Static Strategies for Worksharing with Unrecoverable Interruptions (Extended version)

Anne Benoit, Yves Robert, Arnold Rosenberg, Frédéric Vivien

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Static Strategies for Worksharing with Unrecoverable Interruptions (Extended version)

Anne Benoit,
Yves Robert,
Arnold L. Rosenberg,
Frédéric Vivien

September 2009

Research Report N° 2009-26

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`




Static Strategies for Worksharing with Unrecoverable Interruptions (Extended version)

Anne Benoit, Yves Robert, Arnold L. Rosenberg, Frédéric Vivien

September 2009

Abstract

One has a large workload that is "divisible"—its constituent work's granularity can be adjusted arbitrarily—and one has access to p remote computers that can assist in computing the workload. How can one best utilize these computers? Complicating this question is the fact that each remote computer is subject to interruptions (of known likelihood) that kill all work in progress on it. One wishes to orchestrate sharing the workload with the remote computers in a way that maximizes the expected amount of work completed. Strategies are presented for achieving this goal, by balancing the desire to checkpoint often—thereby decreasing the amount of vulnerable work at any point—vs. the desire to avoid the context-switching required to checkpoint. The current study demonstrates the accessibility of strategies that provably maximize the expected amount of work when there is only one remote computer (the case $p = 1$) and, at least in an asymptotic sense, when there are two remote computers (the case $p = 2$); but the study strongly suggests the intractability of exact maximization for $p \geq 2$ computers. This study responds to that challenge by developing efficient heuristics that employ both checkpointing and work replication as mechanisms for decreasing the impact of work-killing interruptions. The quality of these heuristics, in expected amount of work completed, is assessed through exhaustive simulations that use both idealized models and actual trace data.

Keywords: Fault-tolerance, scheduling, divisible loads, probabilities

Résumé

On dispose d'une large tâche *divisible* et l'on a accès à p ordinateurs distants pour traiter cette tâche. Comment utiliser au mieux ces ordinateurs ? Le problème est d'autant plus compliqué que chaque ordinateur est sujet à des interruptions (de probabilité connue) tuant tout le travail qu'il est en train d'effectuer. On souhaite orchestrer le traitement du travail par les ordinateurs de manière à maximiser l'espérance de la quantité de travail complété. Des stratégies pour atteindre ce but sont présentées. Ces stratégies sont des compromis entre l'envie d'effectuer souvent des sauvegardes (*checkpoint*) —ce qui diminue à tout moment la quantité de travail qui risque d'être perdue— et le désir d'éviter le coût des changements de contexte requis par ces sauvegardes. Cette étude présente des stratégies qui maximisent l'espérance de la quantité de travail fait quand il y a un seul ordinateur (cas $p = 1$) et, au moins d'un point de vue asymptotique, quand il y a deux ordinateurs distants (cas $p = 2$). Mais cette étude suggère l'intractabilité de ce problème pour $p > 2$ ordinateurs. Ce défi est relevé par la définition d'heuristiques efficaces qui emploient sauvegardes et réplications pour minimiser l'impact des interruptions destructrices de travail. La qualité de ces heuristiques, en quantité de travail accompli, est évaluée au moyen de simulations exhaustives utilisant d'une part des modèles idéaux et d'autre part des traces réelles.

Mots-clés: Tolérance aux pannes, ordonanncement, tâches divisibles, probabilités

Contents

1 Introduction

Technological advances and economic constraints have engendered a variety of modern computing platforms that allow a person who has a massive, compute-intensive workload to enlist the help of others' computers in executing the workload. The resulting cooperating computers may belong to a nearby or remote cluster (of "workstations"; cf. [29]), or they could be geographically dispersed computers that are available under one of the increasingly many modalities of Internet-based computing—such as Grid computing (cf. [14, 19, 18]), global computing (cf. [16]), or volunteer computing (cf. [25]). In order to avoid unintended connotations concerning the organization of the remote computers, we avoid evocative terms such as "cluster" or "grid" in favor of the generic "assemblage." Advances in computing power never come without cost. These new platforms add various types of *uncertainty* to the list of concerns that must be addressed when preparing one's computation for allocation to the available computers: notably, computers can slow down unexpectedly, even failing ever to complete allocated work. The current paper follows in the footsteps of sources such as [5, 11, 13, 21, 27, 34], which present analytic studies of algorithmic techniques for coping with uncertainty in computational settings. Whereas most of these sources address the uncertainty of the computers in an assemblage one computer at a time, we attempt here to view the assemblage as a "team" wherein one computer's shortcomings can be compensated for by other computers, most notably by judiciously *replicating work*, i.e., by allocating some work to more than one computer. Such a team-oriented viewpoint has appeared mainly in experimental studies (cf. [24]); ours is the first analytical study to adopt such a point of view.

The problem. We have a large computational workload whose constituent work is *divisible* in the sense that one can partition chuks of work into arbitrary granularities (cf. [12]). We also have access to $p \geq 1$ identical computers to help us compute the workload via *worksharing* (wherein the owner of the workload allocates work to remote computers that are idle; cf. [35]).

> We study *homogeneous* assemblages in the current paper in order to concentrate only on developing technical tools to cope with uncertainty within an assemblage. We hope to focus in later work on the added complexity of coping with uncertainty within a *heterogeneous* assemblage, whose computers may differ in power and speed.

We address here the most draconian type of uncertainty that can plague an assemblage of computers, namely, vulnerability to *unrecoverable interruptions* that cause us to lose all work currently in progress on the interrupted computer. We wish to cope with such interruptions—whether they arise from hardware failures or from a loaned/rented computer's being reclaimed by its owner, as during an episode of *cycle-stealing* (cf. [5, 13, 31, 32, 34]). The scheduling tool that we employ to cope with these interruptions is *work replication*, the allocation of chunks of work to more than one remote computer. The only external resource to help us use this tool judiciously is our assumed access to *a priori* knowledge of the risk of a computer's having been interrupted—which we assume is the same for all computers.[1]

The goal. Our goal is to maximize the *expected amount of work that gets computed by the assemblage of computers*, no matter which, or how many computers get interrupted. Therefore, we implicitly assume that we are dealing with applications for which even partial output is meaningful, e.g., annotation of metagenomics data. In metagenomics annotation, one has a large number of DNA fragments to classify (as belonging to eukaryotes, prokaryotes, etc.); one would rather have all the DNA fragments processed, but the result of the classification is nevertheless meaningful even if the annotation is fragmentary (this just artificially raises the "unknown" category).

Three challenges. The challenges of scheduling our workload on interruptible remote computers can be described in terms of three dilemmas. The first two apply even to each remote computer individually.[2]

[1] As in [13, 31, 34], our scheduling strategies can be adapted to use statistical, rather than exact, knowledge of the risk of interruption—albeit at the cost of weakened performance guarantees.

[2] We put "parallelism" in quotes when stating these dilemmas because remote computers are (usually) not synchronized, so they do not truly operate *in parallel*.

1. **If** we send each remote computer a large amount of work with each transmission,
   then we both decrease the overhead of packaging work-containing messages and maximize the opportunities for "parallelism" within the assemblage of remote computers,
   but we thereby maximize our vulnerability to losing work because of a remote computer's being interrupted.

On the other hand,

2. **If** we send each remote computer a small amount of work with each transmission,
   then we minimize our vulnerability to interruption-induced losses
   but we thereby maximize message overhead and minimize the opportunities for "parallelism" within the assemblage of remote computers.

The third dilemma arises only when there are at least two remote computers.[3]

3. **If** we *replicate work,* by sending the same work to more than one remote computer,
   then we lessen our vulnerability to interruption-induced losses,
   but we thereby minimize both the opportunities for "parallelism" and the expected productivity advantage from having access to the remote computers.

Approaches to the challenges. *(1) "Chunking" our workload.* We cope with the first two dilemmas by sending work allocations to the remote computers as a sequence of *chunks*[4] rather than as a single block to each computer. This approach, which is advocated in [13, 31, 32, 34], allows each computer to *checkpoint* at various times and, thereby, to protect some of its work from the threat of interruption. *(2) Replicating work.* We allocate some chunks to more than one remote computer in order to enhance their chances of being computed successfully. We use work replication judiciously, in deference to the third dilemma.

> Under our model, the risk of a computer's being interrupted increases as the computer operates, whether it works on our computation or not. This assumption models, e.g., interruptions from hardware failures or from returning owners in cycle-stealing scenarios. Thus, certain chunks of our workload are more vulnerable to being interrupted than others. To wit, the first "round" of allocated chunks involves our first use of the remote computers; hence, these chunks are less likely to be interrupted than are the chunks that are allocated in the second "round": the remote computers will have been operating longer by the time the second "round" occurs. In this manner, the second-"round" chunks are less vulnerable than the third-"round" chunks, and so on.

Because communication to remote computers is likely to be costly in time and overhead, we limit such communication by orchestrating work replication in an *a priori,* static manner, rather than dynamically, in response to observed interruptions. While we thereby duplicate work unnecessarily when there are few interruptions among the remote computers, we also thereby prevent *our* computer, which is the server in the studied scenario, from becoming a communication bottleneck when there are many interruptions. Our cost concerns are certainly justified when we access remote computers over the Internet, but also when accessing computers over a variety of common local-area networks (LANs). Moreover, as noted earlier, we get a good "return" from our conservative work replication, by increasing the expected amount of work done by the remote computers.

In summation, we assume: that we know the instantaneous probability that a remote computer will have been interrupted by time t; that this probability is the same for all remote computers;

[3] The pros and cons of work replication are discussed in [24].

[4] We use the generic "chunk" instead of "task" to emphasize tasks' divisibility: by definition, divisible workloads do not have atomic tasks.

that the probability increases with the amount of time that the computer has been available. These assumptions, which we share with [13, 31, 34], seem to be necessary in order to derive scheduling strategies that are *provably* optimal. As suggested in these sources (cf. footnote 1), one can use approximate knowledge of these probabilities, obtained, say, via trace data, but this will clearly weaken our performance claims for our schedules. Also as noted earlier, the challenge of allowing individual computers to have different probabilities must await a sequel to the current study.

Related work. The literature contains relatively few rigorously analyzed scheduling algorithms for interruptible "parallel" computing in assemblages of computers. Among those we know of, only [5, 13, 31, 32, 34] deal with an *adversarial* model of interruptible computing. One finds in [5] a randomized scheduling strategy which, with high probability, completes within a logarithmic factor of the optimal fraction of the initial workload. In [13, 31, 32, 34], the scheduling problem is viewed as a game against a malicious adversary who seeks to interrupt each remote computer in order to kill all work in progress. Among the experimental sources, [39] studies the use of task replication on a heterogeneous desktop grid whose constituent computers may become definitively unavailable; the objective is to eventually process all work. In a similar context, [3] aims at minimizing both the completion time of applications and the amount of resources used.

There is a very large literature on scheduling divisible workloads on assemblages of computers that are not vulnerable to interruption. We refer the reader first to [12] and its myriad intellectual progeny; not all of these sources share the current study's level of detailed argumentation. One finds in [2], its precursor [33], and its accompanying experimental study [1], an intriguing illustration of the dramatic impact on the scheduling problem for heterogeneous assemblages of having to account for the transmission of the output generated by the computation; a different aspect of the same observation is noted in [10]. Significant preliminary results about assemblages in which communication links, as well as constituent computers, are heterogeneous appear in [10]. Several studies focus on scheduling divisible computations but focus on algorithmically simpler computations whose tasks produce no output. A near-optimal algorithm for such scheduling appears in [40] under a simplified model, in which equal-size chunks of work are sent to remote computers at a frequency determined by the computers' powers. The body of work exemplified by [36, 37, 12, 15, 17] and sources cited therein allow heterogeneity in both communication links and computers, but schedule outputless tasks, under a simple communication model. (It is worth noting that one consequence of a linear, rather than affine communication cost model is that it can be advantageous to distribute work in *many* small pieces, rather than in a few large chunks; cf. [37, 40].) A significant study that shares our focus on tasks having equal sizes and complexities, but that allows workstations to redistribute allocated tasks, appears in [7, 9]. Under the assumption of unit computation time per task, these sources craft linear-programming algorithms that optimize the steady-state processing of tasks. The distribution of inputs and subsequent collection of results form an integral part of [2, 10]; these problems are studied as independent topics in [8].

Even the subset of the divisible workload literature that focuses on collective communication in assemblages of computers is enormous. Algorithms for various collective communication operations appear in [6, 22]. One finds in [20] approximation algorithms for a variant of broadcasting under which receipt of the message "triggers" a "personal" computation whose cost is accounted for within the algorithm.

We do not enumerate here the many studies of computation on assemblages of remote computers, which focus either on systems that enable such computation or on specific algorithmic applications. However, we point to [26] as an exemplar of the former type of study and to [38] as an exemplar of the latter.

2 The Technical Framework

We supply the technical details necessary to turn the informal discussion in the Introduction into a framework in which we can develop and rigorously validate scheduling guidelines.

2.1 The Computation and the Computers

We begin with $W_{(\text{ttl})}$ units of divisible work that we wish to execute on an assemblage of $p \geq 1$ identical computers. Each computer is susceptible to *unrecoverable* interruptions that "kill" all work currently in progress on it. All computers share the same instantaneous probability of being interrupted, and this probability increases with the amount of time the computer has been operating (whether working on our computation or not). We know this probability exactly.[5]

> Because we deal with a single computational application and identical computers, we lose no generality by expressing our results in terms of units of work, rather than the execution time of these units. We paraphrase the following explanation from [2], which uses a similar convention.
>
> Our results rely only on the fact that all work units have the same size and complexity: formally, there is a constant $c > 0$ such that executing w units of work takes cw time units. *The work units' (common) complexity can be an arbitrary function of their (common) size:* c is simply the ratio of the fixed size of a work unit to the complexity of executing that amount of work.

As discussed in the Introduction, the danger of losing work in progress when an interruption incurs mandates that we not just divide our workload into $W_{(\text{ttl})}/p$ equal-size chunks and allocate one chunk to each computer in the assemblage. Instead, we "protect" our workload as best we can, by:

- partitioning it into chunks, the unit of work that we allocate to the computers
- prescribing a schedule for allocating chunks to computers
- allocating some chunks to more than one computer, as a divisible-load mode of work replication.

As noted in the Introduction, we treat intercomputer communication as a resource to be used very conservatively—which is certainly justified when communication is over the Internet, and often when communication is over common local-area networks (LANs). Specifically, we try to avoid having *our* computer become a communication bottleneck by orchestrating chunk replications in an *a priori,* static manner—even though this leads to duplicated work when there are few or no interruptions—rather than dynamically, in response to observed interruptions.

2.2 Modeling Interruptions and Expected Work

2.2.1 The interruption model

Within our model, all computers share the same *risk function*, i.e., the same instantaneous probability, $Pr(w)$, of having been interrupted by the end of "the first w time units."

> *Recall that we measure time in terms of work units that* could have been *executed "successfully," i.e., with no interruption. In other words "the first w time units" is the amount of time that a computer* would have needed *to compute w work units* if *it had started working on them when the entire worksharing episode began.*
>
> This time scale is shared by all computers in our homogeneous setting.

Of course, $Pr(w)$ increases with w; we assume that we know its value exactly: see, however, footnote 1.

It is useful in our study to generalize our measure of risk by allowing one to consider many baseline moments. We denote by $Pr(s, w)$ the probability that a computer has not been interrupted during the first s "time units" but has been interrupted by "time" $s + w$. Thus, $Pr(w) = Pr(0, w)$ and $Pr(s, w) = Pr(s + w) - Pr(s)$.

[5] As stated earlier, our analyses can be modified to accommodate probabilities that are known only statistically.

We let[6] $\kappa \in (0,1]$ be a constant that weights our probabilities. We illustrate the role of κ as we introduce two specific common risk functions Pr, the first of which is our focus in the current study.

Linearly increasing risk. The risk function that will be the main focus of our study is $Pr(w) = \kappa w$. It is the most natural model in the absence of further information: the failure risk grows linearly, in proportion to the time spent, or equivalently to the amount of work done. This linear model covers a wide range of cycle-stealing scenarios, but also situations when interruptions are due to hardware failures.

In this case, we have the density function

$$dPr \;=\; \begin{cases} \kappa dt & \text{for } \; t \in [0, 1/\kappa] \\ 0 & \text{otherwise} \end{cases}$$

so that

$$Pr(s,w) \;=\; \min\left\{1,\; \int_s^{s+w} \kappa dt\right\} \;=\; \min\{1, \kappa w\} \tag{1}$$

The constant $1/\kappa$ will recur repeatedly in our analyses, since it can be viewed as the time by which an interruption is certain, i.e., will have occurred with probability 1. To enhance legibility of the rather complicated expressions that populate our analyses, we henceforth denote the quantity $1/\kappa$ by X.

Geometrically decaying lifespan. A commonly studied risk function, which models a variety of common "failure" scenarios, is $Pr(w) = 1 - \exp^{-\kappa w}$, wherein the probability of a computer's surviving for one more "time step" decays geometrically. More precisely,

$$\begin{aligned} Pr(w,1) = Pr(w+1) - Pr(w) \;&=\; (1 - \exp^{-\kappa(w+1)}) - (1 - \exp^{-\kappa w}) \\ &=\; (1 - \exp^{-\kappa}) \exp^{-\kappa w}\,. \end{aligned}$$

One might expect such a risk function, for instance, when interruptions are due to someone's leaving work for the day; the longer s/he is absent, the more likely it is that s/he is gone for the night.

In this case, we have the density function $dPr = \kappa \exp^{-\kappa t}\, dt$, so that

$$Pr(s,w) \;=\; \int_s^{s+w} \kappa \exp^{-\kappa t}\, dt \;=\; \exp^{-\kappa s}(1 - \exp^{-\kappa w}).$$

2.2.2 Expected work production

Risk functions help us finding an efficient way to chunk work for, and allocate work to, the remote computers, in order to maximize the expected work production of the assemblage. To this end, we focus on a workload consisting of $W_{(\text{ttl})}$ work units, and we let $\mathsf{W}_{(\text{cmp})}$ be the random variable whose value is the number of work units that the assemblage executes successfully under a given scheduling regimen. Stated formally, we are striving to maximize the expected value (or, expectation) of $\mathsf{W}_{(\text{cmp})}$.

We perform our study under two models, which play different roles as one contemplates the problem of scheduling a large workload. The models differ in the way chunk execution times relate to chunk sizes. The actual time for processing a chunk of work has several components:

- There is the overhead for transmitting the chunk to the remote computer. This may be a trivial amount of actual time if one must merely set up the communication, or it may be a quite significant amount if one must, say, encode the chunk before transmitting it. In the latter case, the overhead can be proportional to the chunk size.
- There is the time to actually transmitting the chunk, which is proportional to the chunk size.

[6] As usual, $(a,b]$ (resp., $[a,b]$) denotes the real interval $\{x \mid a < x \le b\}$ (resp., $\{x \mid a \le x \le b\}$).

- There is the actual time that the remote computer spends executing the chunk, which, by assumption, is proportional to the chunk size.
- There is the time that the remote computer spends checkpointing after computing a chunk. This may be a trivial amount of actual time—essentially just a context switch—if the chunk creates little output (perhaps just a YES/NO decision), or it may be a quite significant amount if the chunk creates a sizable output (e.g., a matrix inversion).

In short, there are two classes of time-costs, those that are proportional to the size of a chunk and those that are fixed constants. It simplifies our formal analyses to fold the first class of time-costs into a single quantity that is proportional to the size of a chunk and to combine the second class into a single fixed constant. When chunks are large, the second cost will be minuscule compared to the first. This suggests that the fixed costs can be ignored, but one must be careful: if one ignores the fixed costs, then there is no disincentive to, say, deploying the workload to the remote computers in $n+1$ chunks, rather than n. Of course, increasing the number of chunks tends to make chunks smaller—which increases the significance of the second cost! One could, in fact, strive for adaptive schedules that change their strategies depending on the changing ratios between chunk sizes and fixed costs. However, for the reasons discussed earlier, we prefer to seek *static* scheduling strategies, at least until we have a well-understood arsenal of tools for scheduling interruptible divisible workloads. Therefore, we perform the current study with two fixed cost models, striving for optimal schedules under each. (1) The *free-initiation model* is characterized by *not* charging the owner of the workload a per-chunk fixed cost. This model focuses on situations wherein the fixed costs are negligible compared to the chunk-size-dependent costs. (2) The *charged-initiation model,* which more accurately reflects the costs incurred with real computing systems, is characterized by accounting for both the fixed and chunk-size-dependent costs.

The *free-initiation* model. This model, which assesses no per-chunk cost, is much the easier of our two models to analyze. The results obtained using this model approximate reality well when one knows *a priori* that chunks must be large. One situation that mandates large chunks is when communication is over the Internet, so that one must have every remote computer do a substantial amount of the work in order to amortize the time-cost of message transmission (cf. [25]). In such a situation, one will keep chunks large by placing a bound on the number of scheduling "rounds," which counteracts this model's tendency to increase the number of "rounds" without bound. Importantly also: the free-initiation model allows us to obtain predictably good *bounds* on the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under the charged-initiation model, in situations where such bounds are prohibitively hard to derive directly; cf. Theorem 1.

Under the free-initiation model, the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under a given scheduling regimen Σ and for a workload $W_{(\mathrm{ttl})}$, denoted $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma)$, the superscript "f" denoting "free(-initiation)," is

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) \;=\; \int_0^\infty Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u \text{ under } \Sigma)\; du.$$

Let us illustrate this model via three simple calculations of $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma)$. In these calculations, the regimen Σ allocates the whole workload and deploys it on a single computer. To enhance legibility, let the phrase "under Σ" within "$Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u$ under $\Sigma)$" be specified implicitly by context.

Deploying the workload as a single chunk. Under regimen Σ_1 the whole workload is deployed as a single chunk on a single computer. By definition, $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1)$ for an arbitrary risk function Pr is given by

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1) \;=\; W_{(\mathrm{ttl})}\left(1 - Pr(W_{(\mathrm{ttl})})\right). \tag{2}$$

Deploying the workload in two chunks. Regimen Σ_2 specifies how the workload is split into the two chunks of respective sizes $\omega_1 > 0$ and $\omega_2 > 0$, where $\omega_1 + \omega_2 = W_{(\mathrm{ttl})}$. The following derivation

determines $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2)$ for an arbitrary risk function Pr.

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2) &= \int_0^{\omega_1} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du + \int_{\omega_1}^{\omega_1+\omega_2} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \\ &= \int_0^{\omega_1} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq \omega_1)du + \int_{\omega_1}^{\omega_1+\omega_2} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq \omega_1+\omega_2)du \\ &= \omega_1(1 - Pr(\omega_1)) \ + \ \omega_2(1 - Pr(\omega_1+\omega_2)). \end{aligned} \tag{3}$$

Deploying the workload in n chunks. Continuing the reasoning of the cases $n = 1$ and $n = 2$, we finally obtain the following general expression for expectation $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$ for an arbitrary risk function Pr, when Σ_n partitions the whole workload into n chunks of respective sizes $\omega_1 > 0$, $\omega_2 > 0$, $\ldots$, $\omega_n > 0$ such that $\omega_1 + \cdots + \omega_n = W_{(\mathrm{ttl})}$.

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n) &= \int_0^{\omega_1} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du + \int_{\omega_1}^{\omega_1+\omega_2} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \\ &\quad + \cdots + \int_{\omega_1+\cdots+\omega_{n-1}}^{\omega_1+\cdots+\omega_{n-1}+\omega_n} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u)du \\ &= \int_0^{\omega_1} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq \omega_1)du \\ &\quad + \int_{\omega_1}^{\omega_1+\omega_2} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq \omega_1+\omega_2)du \qquad (4) \\ &\quad + \cdots + \int_{\omega_1+\cdots+\omega_{n-1}}^{\omega_1+\cdots+\omega_{n-1}+\omega_n} Pr(\mathsf{W}_{(\mathrm{cmp})} \geq \omega_1+\cdots+\omega_n)du \\ &= \omega_1(1 - Pr(\omega_1)) \ + \ \omega_2(1 - Pr(\omega_1+\omega_2)) \qquad (5) \\ &\quad + \cdots + \ \omega_n(1 - Pr(\omega_1+\cdots+\omega_n)). \end{aligned}$$

Optimizing expected work-production on one remote computer. One goal of our study is to learn how to craft, for each integer n, a scheduling regimen Σ that maximizes $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma)$. However, we have a more ambitious goal, which is motivated by the following observation.

Many risk functions—such as the linear risk function—represent situations wherein the remote computers are *certain* to have been interrupted no later than a known eventual time. In such a situation, one might get more work done, in expectation, by not deploying the entire workload: one could increase the expectation by making the last deployed chunk even a tiny bit smaller than needed to deploy all $W_{(\mathrm{ttl})}$ units of work.

> We shall see the preceding observation in operation in Theorem 2 for the free-initiation model and in Theorem 3 for the charged-initiation model.

Thus, our ultimate goal when considering a single remote computer (the case $p = 1$), is to determine, for each integer n:

- how to select n chunk sizes that collectively sum to *at most* $W_{(\mathrm{ttl})}$ (rather than to exactly $W_{(\mathrm{ttl})}$ as in the preceding paragraphs),
- how to select n chunks of these sizes out of our workload,
- how to schedule the deployment of these chunks

in a way that maximizes the expected amount of work that gets done. We formalize this goal via the function $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n)$:

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n) \ = \ \max\{\omega_1(1 - Pr(\omega_1)) + \cdots + \omega_n(1 - Pr(\omega_1+\cdots+\omega_n))\},$$

where the maximization is over all n-tuples of positive chunk sizes that sum to at most $W_{(\mathrm{ttl})}$:

$$\{\omega_1 \geq 0, \ \omega_2 \geq 0, \ldots, \ \omega_n \geq 0\} \quad \text{such that} \quad \omega_1 + \omega_2 + \cdots + \omega_n \leq W_{(\mathrm{ttl})}$$

The *charged-initiation* model. This model is much harder to analyze than the free-initiation model, even when there is only one remote computer. In compensation, *the charged-initiation model often allows one to determine analytically the best numbers of chunks and of "rounds"* (when there are multiple remote computers). Under this model, the overhead for each additional chunk is a fixed cost—which, in common with time, we measure in units of work—that is added to the cost of computing of each chunk; we denote this overhead by ε (for instance this may correspond to a checkpointing cost). Under this model, the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under a given scheduling regimen Σ and for a worload $W_{(\mathrm{ttl})}$, denoted $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma)$, the superscript "c" denoting "charged(-initiation)" is

$$E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma) \;=\; \int_0^\infty Pr(\mathsf{W}_{(\mathrm{cmp})} \geq u + \varepsilon)\; du.$$

We find that, when the whole workload is deployed as a single chunk,

$$E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma_1) \;= W_{(\mathrm{ttl})}\left(1 - Pr(W_{(\mathrm{ttl})} + \varepsilon)\right),$$

and when work is deployed as two chunks of respective sizes ω_1 and ω_2,

$$E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma_2) \;= \omega_1(1 - Pr(\omega_1 + \varepsilon)) + \omega_2(1 - Pr(\omega_1 + \omega_2 + 2\varepsilon)).$$

Finally, we let $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, k)$ be the analogue for the charged-initiation model of the parameterized free-initiation expectation $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, k)$

Relating the two models. One can bound the work completed under the charged-initiation model via the free-initiation model. This justifies our primary focus on the free-initiation model.

Theorem 1. (Charged-initiation output vs. Free-initiation output)
Let $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n)$ and $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n)$ denote, respectively, the optimal n-chunk expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under the charged-*initiation model and under the* free-*initiation model. Then:*

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n) \;\geq\; E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n) \;\geq\; E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n) - n\varepsilon. \tag{6}$$

Proof. The lefthand bound in (6) is obvious, because risk functions are nondecreasing —so that, for any given scheduling regimen, the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under the charged-initiation model cannot exceed the expected value under the free-initiation model.

To derive the righthand bound in (6), let us focus on any optimal scheduling regimen Σ under the free-initiation model with p remote computers. Σ schedules the load via n chunks $\mathcal{W}_1, \ldots, \mathcal{W}_n$ of size $\omega_1 > 0, \ldots, \omega_n > 0$. We note $W_{(\mathrm{dpl})} = \cup_{i=1}^n \mathcal{W}_i$. For any $j \in [1, p]$, $\Sigma(j, k)$ denotes the k-th chunk executed on computer j by schedule regimen Σ. In other words, computer j executes chunks in the order $\Sigma(j, 1), \Sigma(j, 2), \ldots, \Sigma(j, n)$.[7]

Note that, because we target here a p-computer schedule, two different chunks may contain some shared piece of work: whatever i and j in $[1, n]$, we may have $i \neq j$ and $\mathcal{W}_i \cap \mathcal{W}_j \neq \emptyset$. We then partition $W_{(\mathrm{dpl})}$ into $\mathcal{X} = \{\mathcal{X}_1, ..., \mathcal{X}_m\}$ such that, for any partition-element $\mathcal{X}_i$ and any chunk $\mathcal{W}_j$, either $\mathcal{X}_i$ is included in $\mathcal{W}_j$ ($\mathcal{X}_i \subset \mathcal{W}_j$), or $\mathcal{X}_i$ and $\mathcal{W}_j$ have no elements in common ($\mathcal{X}_i \cap \mathcal{W}_j = \emptyset$). Then, under scheduling regimen Σ, any computer is attempting to process the whole partition-element $\mathcal{X}_i$ or is not attempting to process any part of it. Then π_i is the subset of the computers on which $\mathcal{X}_i$ is scheduled under Σ. If $j \in \pi_i$, $\sigma(j, i)$ is the rank of the first chunk scheduled on computer j and containing $\mathcal{X}_i$; formally $\sigma(j, i) = \min\{k | \mathcal{X}_i \subset \Sigma(j, k)\}$.

From schedule Σ we define a new schedule Σ'. For any i in $[1, n]$, let $\omega_i' = \max\{0, \omega_i - \varepsilon\}$, and let $\mathcal{W}_i'$ be any subset of size ω_i' of $\mathcal{W}_i$. Σ' denotes the scheduling regimen that executes the chunks $\mathcal{W}_1', \ldots, \mathcal{W}_n'$ on the p-computers exactly as Σ executes the chunks $\mathcal{W}_1, \ldots, \mathcal{W}_n$ on those same computers, except that zero-length chunks are not executed but skipped (in order not to pay the charged-initiation for nothing). We account for these zero-length chunks in the following equations, via the function

$$\mathbb{1}_{\omega_i'} \;=\; \begin{cases} 1 & \text{if } \omega_i' \neq 0 \\ 0 & \text{if } \omega_i' = 0. \end{cases}$$

[7] Without loss of generality we can assume that each computer executes the n chunks. Indeed, we never decrease the expectation of a schedule by extending it so that each computer attempts to execute each chunk.

We then define the objects $\mathcal{X}'_i$, π'_i and $\sigma'(j,i)$ as we defined $\mathcal{X}_i$, π_i and $\sigma(j,i)$ except that we further impose that any partition-elements $\mathcal{X}'_i$ be a subset of some partition-element $\mathcal{X}_j$ (we thus subdivide the partition $\mathcal{X}$ to obtain the partition $\mathcal{X}'$). Then, $\mathcal{X}_{\tau(i)}$ is the element of $\mathcal{X}$ containing $\mathcal{X}'_i$. Finally, let $\mathcal{I}'$ be the largest subset of $\mathcal{X}'$ satisfying the property:

$$\forall i \in \mathcal{I}',\ \{j \mid \mathcal{X}'_i \subset \mathcal{W}'_j\} = \{j \mid \mathcal{X}_{\tau(i)} \subset \mathcal{W}_j\}.$$

Informally speaking, the pieces of work in an element of $\mathcal{I}'$ belong to the same chunks under the two schedules Σ and Σ'. If $\mathcal{X}'_i$ does not belong to $\mathcal{I}'$, this means that there exist some chunk $\mathcal{W}_j$ such that some piece of work in $\mathcal{X}_{\tau(i)}$ belongs to $\mathcal{W}_j$ but not to $\mathcal{W}'_j$. Then, $\mathcal{X}'_i \subset \mathcal{W}_j \setminus \mathcal{W}'_j$, $\cup_{i\notin\mathcal{I}'}\mathcal{X}'_i \subset \cup_{i=1}^{n}\mathcal{W}_i \setminus \mathcal{W}'_j$ and:

$$\sum_{i\notin\mathcal{I}'} |\mathcal{X}'_i| = \left|\bigcup_{i\notin\mathcal{I}'} \mathcal{X}'_i\right| \leq \left|\bigcup_{i=1}^{n} \mathcal{W}_i \setminus \mathcal{W}'_j\right| \leq \sum_{i=1}^{n} |\mathcal{W}_i \setminus \mathcal{W}'_j| = n\varepsilon.$$

Since Σ' implicitly specifies a scheduling regimen for the charged-initiation model when using $\leq n$ chunks, the expected value of $\mathsf{W}_{(\mathrm{cmp})}$ under Σ' obviously cannot exceed the expected value under the *best* scheduling regimen for the charged-initiation model when using $\leq n$ chunks. Therefore,

$$\begin{aligned}
E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n) &\geq E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, \Sigma') \\
&= \sum_{i=1}^{m'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi'_i} Pr\left(\sum_{k=1}^{\sigma'(j,i)} (\omega'_{\Sigma(j,k)} + \varepsilon \mathbb{1}_{\omega'_{\Sigma(j,k)}})\right)\right) \\
&= \sum_{i=1}^{m'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi'_i} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \mathbb{1}_{\omega'_{\Sigma(j,k)}}(\omega'_{\Sigma(j,k)} + \varepsilon)\right)\right) \\
&= \sum_{i=1}^{m'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi'_i} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \mathbb{1}_{\omega'_{\Sigma(j,k)}}\omega_{\Sigma(j,k)}\right)\right) \\
&\geq \sum_{i=1}^{m'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi'_i} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \omega_{\Sigma(j,k)}\right)\right) \\
&\geq \sum_{i\in\mathcal{I}'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi'_i} Pr\left(\sum_{k=1}^{\sigma'(j,i)} \omega_{\Sigma(j,k)}\right)\right) \\
&= \sum_{i\in\mathcal{I}'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi_{\tau(i)}} Pr\left(\sum_{k=1}^{\sigma(j,\tau(i))} \omega_{\Sigma(j,k)}\right)\right) \\
&= E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) - \sum_{i\notin\mathcal{I}'} |\mathcal{X}'_i| \left(1 - \prod_{j\in\pi_{\tau(i)}} Pr\left(\sum_{k=1}^{\sigma(j,\tau(i))} \omega_{\Sigma(j,k)}\right)\right) \\
&\geq E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) - \sum_{i\notin\mathcal{I}'} |\mathcal{X}'_i| \\
&= E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma) - \left|\bigcup_{i\notin\mathcal{I}'} \mathcal{X}'_i\right| \\
&\geq E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n) - n\varepsilon
\end{aligned}$$

which yields the righthand bound. □

3 Scheduling for a Single Remote Computer

This section is devoted to studying how to schedule optimally when there is only a single remote computer that is subject to the linear risk of interruption: $Pr(w) = \min(1, \kappa w)$. Some of the results we derive bear a striking similarity to their analogues in [13], despite certain substantive differences in models.

3.1 An Optimal Schedule under the Free-Initiation Model

We begin with a simple illustration of why the risk of losing work because of an interruption must affect our scheduling strategy, even when there is only one remote computer and even when dispatching a new chunk of work incurs no cost, *i.e.,* under the free-initiation model.

When the amount of work $W_{(\mathrm{ttl})}$ is no larger than X (recall that $\kappa = 1/X$ is the constant that accounts for the size of the work-unit), then instantiating the linear risk function, $Pr(W_{(\mathrm{ttl})}) = \kappa W_{(\mathrm{ttl})}$, in (2) shows that the expected amount of work achieved when deploying the entire workload in a single chunk is

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1) \;=\; W_{(\mathrm{ttl})} - \kappa W_{(\mathrm{ttl})}^2.$$

Similarly, instantiating this risk function in (3) shows that the expected amount of work achieved when deploying the entire workload using two chunks, of respective sizes $\omega_1 > 0$ and $\omega_2 > 0$ is (recalling that $\omega_1 + \omega_2 = W_{(\mathrm{ttl})}$)

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2) &= \omega_1(1 - \omega_1\kappa) + \omega_2(1 - (\omega_1 + \omega_2)\kappa)) \\ &= W_{(\mathrm{ttl})} - (\omega_1^2 + \omega_1\omega_2 + \omega_2^2)\kappa \\ &= W_{(\mathrm{ttl})} - W_{(\mathrm{ttl})}^2\kappa + \omega_1\omega_2\kappa. \end{aligned}$$

We observe that

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_2) - E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_1) \;=\; \omega_1\omega_2\kappa \;>\; 0.$$

Thus, as one would expect intuitively: *For any fixed total workload, one increases the expectation of* $\mathsf{W}_{(\mathrm{cmp})}$ *by deploying the workload as two chunks, rather than one—no matter how one sizes the chunks.*

Continuing with the preceding reasoning, we can actually characterize the optimal—i.e., expectation-maximizing—schedule for any fixed number of chunks. (We thereby also identify a weakness of the free-initiation model: increasing the number of chunks always increases the expected amount of work done—so the (unachievable) "optimal" strategy would deploy infinitely many infinitely small chunks.)

Theorem 2. (One remote computer: free-initiation model)
Say that one wishes to deploy $W_{(\mathrm{ttl})} \in [0, X]$ *units of work to a single remote computer in at most* n *chunks, for some positive integer* n*. In order to maximize the expectation of* $\mathsf{W}_{(\mathrm{cmp})}$*, one should have all* n *chunks share the same size, namely,* Z/n *units of work, where*

$$Z \;=\; \min\left\{W_{(\mathrm{ttl})}, \frac{n}{n+1}X\right\}.$$

In expectation, this optimal schedule completes

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n) \;=\; Z \;-\; \frac{n+1}{2n}Z^2\kappa$$

units of work.

Note that for fixed $W_{(\mathrm{ttl})}$, $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n)$ increases with n.

Proof. Let us partition the $W_{(\mathrm{ttl})}$-unit workload into $n+1$ chunks, of respective sizes $\omega_1 \geq 0, \ldots,$ $\omega_n \geq 0$, $\omega_{n+1} \geq 0$, with the intention of deploying the first n of these chunks.

(a) Our assigning the first n chunks *nonnegative,* rather than *positive* sizes affords us a convenient way to talk about "at most n chunks" using only the single parameter n. (b) By creating $n+1$ chunks rather than n, we allow ourselves to hold back some work in order to avoid what would be a certain interruption of the nth chunk. Formally, exercising this option means making ω_{n+1} positive; declining the option—thereby deploying all $W_{(\mathrm{ttl})}$ units of work—means setting $\omega_{n+1} = 0$.

Each specific such partition specifies an n-chunk schedule Σ_n. Our challenge is to choose the sizes of the $n+1$ chunks in a way that maximizes $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$. To simplify notation, let $Z = \omega_1 + \cdots + \omega_n$ denote the portion of the entire workload that we actually deploy.

Extending the reasoning from the cases $n = 1$ and $n = 2$, one obtains easily from (5) the expression

$$\begin{aligned} E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n) &= \omega_1(1-\omega_1\kappa) + \omega_2(1-(\omega_1+\omega_2)\kappa) + \cdots + \\ &\quad + \cdots + \omega_n(1-(\omega_1+\cdots+\omega_n)\kappa) \qquad (7) \\ &= Z - Z^2\kappa + \left[\sum_{1\le i<j\le n} \omega_i\omega_j\right]\kappa. \qquad (8) \end{aligned}$$

Standard arguments show that the bracketed sum in (8) is maximized when all ω_i's share the common value Z/n, in which case, the sum achieves the value $\frac{1}{n^2}\binom{n}{2}Z^2\kappa$. Since maximizing the sum also maximizes $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$, simple arithmetic yields:

$$E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n) = Z - \frac{n+1}{2n}Z^2\kappa.$$

Viewing this expression for $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$ as a function of Z, we note that the function is unimodal, increasing until $Z = \dfrac{n}{(n+1)\kappa}$ and decreasing thereafter. Setting this value for Z, gives us the maximum value for $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, \Sigma_n)$, i.e., the value of $E^{(\mathrm{f})}(W_{(\mathrm{ttl})}, n)$. The theorem follows. □

3.2 An Optimal Schedule under the Charged-Initiation Model

Under the *charged-initiation* model—i.e., on a computing platform wherein processing a new chunk of work (for transmission or checkpointing) *does* incur a cost (that we must account for)—deriving the optimal strategy becomes dramatically more difficult, even when there is only one remote computer and even when we know *a priori* how many chunks we wish to employ.

Theorem 3. ***(One remote computer: charged-initiation model)*** *Say that one wishes to deploy $W_{(\mathrm{ttl})} \in [0, X]$ units of work, where $X \ge \varepsilon$, to a single remote computer in at most n chunks, for some positive integer n. Let $n_1 = \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon}-1\right)\right\rfloor$ and $n_2 = \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\mathrm{ttl})}/\varepsilon}+1\right)\right\rfloor$. The unique regimen for maximizing $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n)$ specifies $m = \min\{n, n_1, n_2\}$ chunks: the first has size*[8]

$$\omega_{1,m} = \frac{W_{(\mathrm{dpl})}}{m} + \frac{m-1}{2}\varepsilon$$

where

$$W_{(\mathrm{dpl})} = \min\left\{W_{(\mathrm{ttl})},\ \frac{m}{m+1}X - \frac{m}{2}\varepsilon\right\}; \qquad (9)$$

and the $(i+1)$th chunk inductively has size

$$\omega_{i+1,m} = \omega_{i,m} - \varepsilon.$$

[8]The second subscript of ω reminds us how many chunks the workload is divided into.

In expectation, this schedule completes

$$E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n) \;=\; W_{(\mathrm{dpl})} - \frac{m+1}{2m} W_{(\mathrm{dpl})}^2 \kappa - \frac{m+1}{2} W_{(\mathrm{dpl})} \varepsilon\kappa + \frac{(m-1)m(m+1)}{24} \varepsilon^2 \kappa \quad (10)$$

units of work.

Note that $E^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n)$ is maximal for any value of n no smaller than $\min\{n_1, n_2\}$.

Proof. We proceed by induction on the number n of chunks we want to partition our $W_{(\mathrm{ttl})}$ units of work into. We denote by $\mathcal{E}_{\mathrm{opt}}^{(\mathrm{c})}(W_{(\mathrm{ttl})}, n)$ the maximum expected amount of work that a schedule can complete under such a partition.

Focus first on the case $n = 1$. When work is allocated in a single chunk, the maximum expected amount of total work completed is, by definition:

$$\mathcal{E}_{\mathrm{opt}}^{(\mathrm{c})}(W_{(\mathrm{ttl})}, 1) \;=\; \max_{0 \le \omega_{1,1} \le W_{(\mathrm{ttl})}} \mathcal{E}^{(\mathrm{c})}(\omega_{1,1}) \quad \text{where}$$

$$\mathcal{E}^{(\mathrm{c})}(\omega_{1,1}) \;=\; \max_{0 \le \omega_{1,1} \le W_{(\mathrm{ttl})}} \omega_{1,1}(1 - (\omega_{1,1} + \varepsilon)\kappa).$$

We determine the optimal size of $\omega_{1,1}$ by viewing this quantity as a variable in the closed interval $[0, W_{(\mathrm{ttl})}]$ and maximizing $\mathcal{E}^{(\mathrm{c})}(\omega_{1,1})$ symbolically. We thereby find that $\mathcal{E}^{(\mathrm{c})}(\omega_{1,1})$ is maximized by setting

$$\omega_{1,1} \;=\; \min\left\{W_{(\mathrm{ttl})}, \frac{1}{2\kappa} - \frac{\varepsilon}{2}\right\},$$

so that

$$\mathcal{E}_{\mathrm{opt}}^{(\mathrm{c})}(W_{(\mathrm{ttl})}, 1) \;=\; \begin{cases} \dfrac{1}{4\kappa} - \dfrac{\varepsilon}{2} + \dfrac{\varepsilon^2}{4}\kappa & \text{if } W_{(\mathrm{ttl})} > \dfrac{1}{2\kappa} - \dfrac{\varepsilon}{2}, \\ W_{(\mathrm{ttl})} - W_{(\mathrm{ttl})}^2 \kappa - W_{(\mathrm{ttl})} \varepsilon\kappa & \text{otherwise.} \end{cases}$$

(Note that $\omega_{1,1}$ has a non-negative size because of the natural hypothesis that $X \ge \varepsilon$.)

We now proceed to general values of n by induction. We begin by assuming that the conclusions of the theorem have been established for the case when the workload is split into $n \ge 1$ positive-size chunks. We also assume that n is no greater than $\min\{n_1, n_2\}$. In other words, we assume that any optimal solution with at most n chunks used n positive-size chunks.

As our first step in analyzing how best to deploy $n + 1$ positive-size chunks, we note that the only influence the first n chunks of work have on the probability that the last chunk will be computed successfully is in terms of their cumulative size.

> Let us clarify this last point, which follows from the failure probability model. Denote by A_n the cumulative size of the first n chunks of work in the expectation-maximizing $(n + 1)$-chunk scenario; i.e., $A_n = \sum_{i=1}^{n} \omega_{i,n+1}$. Once A_n is specified, the probability that the remote computer will be interrupted while working on the $(n + 1)$th chunk depends only on the value of A_n, *not* on the way the A_n units of work have been divided into chunks.

This fact means that once one has specified the cumulative size of the workload that comprises the first n chunks, the best way to partition this workload into chunks is as though it were the only work in the system, i.e., as if there were no $(n+1)$th chunk to be allocated. Thus, one can express $\mathcal{E}_{\mathrm{opt}}(W_{(\mathrm{ttl})}, n + 1)$ in terms of A_n (whose value must, of course, be determined) and $\mathcal{E}_{\mathrm{opt}}(A_n, n)$, via the following maximization.

$$\mathcal{E}_{\mathrm{opt}}(W_{(\mathrm{ttl})}, n+1) \;=\; \max\left\{\mathcal{E}_{\mathrm{opt}}(A_n, n) \;+\; \omega_{n+1,n+1}\left(1 - (A_n + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right)\right\},$$

where the maximization is over all values for A_n in which

$$\begin{array}{rll} A_n & > 0 & \text{allowing for the } n \text{ previous chunks} \\ \omega_{n+1,n+1} & \geq 0 & \text{allowing for an } (n+1)\text{th chunk} \\ A_n + \omega_{n+1,n+1} & \leq W_{(\text{ttl})} & \text{because the total workload has size } W_{(\text{ttl})} \\ A_n + \omega_{n+1,n+1} + (n+1)\varepsilon & \leq X & \text{reflecting the risk and cost models} \end{array}$$

The last of these inequalities acknowledges that the remote computer is certain to be interrupted (with probability 1) before it can complete the $(n+1)$th chunk of work, if its overall workload is no smaller than $X - (n+1)\varepsilon$.

We now have two cases to consider, depending on the size of A_n.

Case 1: $A_n < \dfrac{n}{n+1}X - \dfrac{n}{2}\varepsilon$.

By assumption, the expectation-maximizing regimen deploys A_n units of work via its first n chunks. By induction, expression (10) tells us that the expected amount of work completed by deploying these A_n units is

$$\mathcal{E}^{(c)}_{\text{opt}}(A_n, n) \;=\; A_n - \frac{n+1}{2n}A_n^2\kappa - \frac{n+1}{2}A_n\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa.$$

Let $W_{(\text{dpl})}$ denote the total work that is actually allocated: $W_{(\text{dpl})} = A_n + \omega_{n+1,n+1}$. In the following calculations, we write $\omega_{n+1,n+1}$ as $W_{(\text{dpl})} - A_n$, in order to represent the $(n+1)$-chunk scenario entirely via quantities that arise in the n-chunk scenario.

We focus on

$$\begin{aligned} \mathcal{E}^{(1)}(A_n, \omega_{n+1,n+1}) &= \mathcal{E}_{\text{opt}}(A_n, n) + (W_{(\text{dpl})} - A_n)\left(1 - \left(W_{(\text{dpl})} + (n+1)\varepsilon\right)\kappa\right) \\ &= \left(A_n - \frac{n+1}{2n}A_n^2\kappa - \frac{n+1}{2}A_n\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa\right) \\ &\quad + (W_{(\text{dpl})} - A_n)\left(1 - \left(W_{(\text{dpl})} + (n+1)\varepsilon\right)\kappa\right) \\ &= \left(W_{(\text{dpl})} + \frac{n+1}{2}\varepsilon\right)A_n\kappa - \frac{n+1}{2n}A_n^2\kappa \\ &\quad + W_{(\text{dpl})}(1 - (W_{(\text{dpl})} + (n+1)\varepsilon)\kappa) + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa. \end{aligned}$$

For a given value of $W_{(\text{dpl})}$, we look for the best value for A_n using the preceding expression. We note first that

$$\frac{\partial \mathcal{E}^{(1)}(A_n, \omega_{n+1,n+1})}{\partial A_n} \;=\; -\frac{n+1}{n}A_n\kappa + W_{(\text{dpl})}\kappa + \frac{n+1}{2}\varepsilon\kappa.$$

We note next that, for fixed $W_{(\text{dpl})}$, the quantity $\mathcal{E}^{(1)}(A_n, \omega_{n+1,n+1})$ begins to increase with A_n and then decreases. The value for A_n that maximizes this expectation, which we denote $A_n^{(\text{opt})}$, is

$$A_n^{(\text{opt})} \;=\; \min\left\{W_{(\text{ttl})}, \frac{n}{n+1}W_{(\text{dpl})} + \frac{n}{2}\varepsilon\right\}.$$

When $W_{(\text{ttl})} \leq (n/(n+1))W_{(\text{dpl})} + \frac{1}{2}n\varepsilon$, $A_n^{(\text{opt})} = W_{(\text{ttl})}$, meaning that the $(n+1)$th chunk is empty, and the schedule does *not* optimize the expected work. (In the charged-initiation model an empty chunk decreases the overall probability). Consequently, we focus *for the moment* on the case

$$A_n^{(\text{opt})} \;=\; \frac{n}{n+1}W_{(\text{dpl})} + \frac{n}{2}\varepsilon \tag{11}$$

(thereby assuming that $W_{(\text{ttl})} \geq (n/(n+1))W_{(\text{dpl})} + \frac{1}{2}n\varepsilon$). Therefore, we have

$$\begin{aligned} \mathcal{E}^{(1)}(A_n^{(\text{opt})}, \omega_{n+1,n+1}) \;=\; & \\ & -\frac{n+2}{2(n+1)}W_{(\text{dpl})}^2\kappa + W_{(\text{dpl})} - \frac{n+2}{2}\varepsilon W_{(\text{dpl})}\kappa + \frac{n(n+1)(n+2)}{24}\varepsilon^2\kappa. \end{aligned}$$

We maximize $\mathcal{E}^{(1)}(A_n^{(\text{opt})}, \omega_{n+1,n+1})$ via the preceding expression by viewing the expectation as a function of $W_{(\text{dpl})}$. We discover that $\mathcal{E}^{(1)}(A_n^{(\text{opt})}, \omega_{n+1,n+1})$ is maximized when

$$W_{(\text{dpl})} \;=\; Z^{(\text{opt})} \;=\; \min\left\{\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,\; W_{(\text{ttl})}\right\}. \tag{12}$$

For this case to be meaningful, the $(n+1)$th chunk must be nonempty, so that $A_n^{(\text{opt})} < Z$; i.e., $Z > \frac{1}{2}n(n+1)\varepsilon$. Therefore, we must simultaneously have:

1. $(n+1)/(n+2)X - \frac{1}{2}(n+1)\varepsilon > \frac{1}{2}n(n+1)\varepsilon$, so that $X > \frac{1}{2}(n+1)(n+2)\varepsilon$, which requires that $n \le \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon} - 3\right)\right\rfloor$.
2. $W_{(\text{ttl})} > \frac{1}{2}n(n+1)\varepsilon$, which requires that $n \le \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\text{ttl})}/\varepsilon} - 1\right)\right\rfloor$.

We can now check the sanity of the result.

$$Z^{(\text{opt})} + (n+1)\varepsilon \;\le\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon + (n+1)\varepsilon \;<\; X,$$

because of the just established condition $\frac{1}{2}(n+1)(n+2)\varepsilon < X$. We also have,

$$A_n^{(\text{opt})} \;=\; \frac{n}{n+1}W_{(\text{dpl})} + \frac{n}{2}\varepsilon \le \frac{n}{n+1}\left(\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon\right) + \frac{n}{2}\varepsilon = \frac{n}{n+2}X < \frac{n}{n+1}X - \frac{n}{2}\varepsilon$$

because $\frac{1}{2}(n+1)(n+2)\varepsilon < X$. Therefore, the solution is consistent with the defining hypothesis for this case—namely, that $A_n < \frac{n}{n+1}X - \frac{n}{2}\varepsilon$.

Before moving on to case 2, we note that the value (11) does, indeed, extend our inductive hypothesis. To wit, the optimal total amount of allocated work, $Z^{(\text{opt})}$, has precisely the predicted value, and the sizes of the first n chunks do follow a decreasing arithmetic progression with common difference ε (by using the induction hypothesis). Finally, the last chunk has the claimed size:

$$\omega_{n+1,n+1} \;=\; Z^{(\text{opt})} - A_n^{(\text{opt})} \;=\; \frac{1}{n+1}Z^{(\text{opt})} - \frac{n}{2}\varepsilon.$$

We turn now to our remaining chores. We must derive the expectation-maximizing chunk sizes for the second case, wherein A_n is "big." And, we must show that the maximal expected work completion in this second case is always dominated by the solution of the first case—which will lead us to conclude that the regimen of the theorem is, indeed, optimal.

Case 2: $A_n \ge \frac{n}{n+1}X - \frac{n}{2}\varepsilon$.

By (9), if the current case's restriction on A_n is an *in*equality, then A_n *cannot* be an optimal cumulative n-chunk work allocation. We lose no generality, therefore, by focusing only on the subcase when the defining restriction of A_n is an *equality:*

$$A_n \;=\; \frac{n}{n+1}X - \frac{n}{2}\varepsilon.$$

For this value of A_n, call it $A_n^\star$, we have

$$\mathcal{E}_{\text{opt}}(W_{(\text{ttl})}, n+1) \;= \max\Big(\mathcal{E}_{\text{opt}}(A_n^\star, n) + \omega_{n+1,n+1}\left(1 - (A_n^\star + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right)\Big),$$

where the maximization is over all values of $\omega_{n+1,n+1}$ in the closed interval $[0,\; W_{(\text{ttl})} - A_n^\star]$.

To determine a value of $\omega_{n+1,n+1}$ that maximizes $\mathcal{E}_{\text{opt}}(W_{(\text{ttl})}, n+1)$ for $A_n^\star$, we focus on the function

$$\begin{aligned}\mathcal{E}^{(2)}(A_n,&\omega_{n+1,n+1})\\ &= \mathcal{E}_{\text{opt}}(A_n, n) + \omega_{n+1,n+1}\left(1 - (A_n + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right)\\ &= \left(A_n - \frac{n+1}{2n}A_n^2\kappa - \frac{n+1}{2}A_n\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa\right)\\ &\quad + \omega_{n+1,n+1}\left(1 - (A_n + \omega_{n+1,n+1} + (n+1)\varepsilon)\,\kappa\right)\\ &= -\kappa\omega_{n+1,n+1}^2 + \left(-\frac{n+2}{2}\kappa\varepsilon + \frac{1}{n+1}\right)\omega_{n+1,n+1}\\ &\quad + \frac{n\left((n+1)^2(n+2)\varepsilon^2\kappa^2 - 12(n+1)\varepsilon\kappa + 12\right)}{24(n+1)\kappa}.\end{aligned}$$

Easily,

$$\frac{\partial\mathcal{E}^{(2)}(A_n, \omega_{n+1,n+1})}{\partial\omega_{n+1,n+1}} \;=\; -2\kappa\omega_{n+1,n+1} - \frac{n+2}{2}\kappa\varepsilon + \frac{1}{n+1}.$$

Knowing $A_n^\star$ exactly, we infer that the value of $\omega_{n+1,n+1}$ that maximizes the expectation $\mathcal{E}^{(2)}(A_n^\star, \omega_{n+1,n+1})$ is

$$\omega_{n+1,n+1} \;=\; \min\left\{\frac{1}{2(n+1)}X - \frac{1}{4}(n+2)\varepsilon,\; W_{(\text{ttl})} - \frac{n}{n+1}X - \frac{n}{2}\varepsilon\right\}.$$

The second term dominates this minimization whenever

$$W_{(\text{ttl})} \;\geq\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon;$$

therefore, if $W_{(\text{ttl})}$ is large enough—as delimited by the preceding inequality—then

$$\mathcal{E}^{(2)}(A_n^\star, \omega_{n+1,n+1}) \;= \frac{2n^2+2n+1}{4(n+1)^2}X - \frac{2n^2+3n+2}{4(n+1)}\varepsilon + \frac{(n+2)(2n^2+5n+6}{48}\kappa\varepsilon^2,$$

When $W_{(\text{ttl})}$ does not achieve this threshold, then

$$\begin{aligned}\mathcal{E}^{(2)}(A_n^\star, \omega_{n+1,n+1}) =& -W_{(\text{ttl})}^2\kappa + \left(\frac{n-2}{2}\kappa\varepsilon + \frac{2n+1}{n+1}\right)W_{(\text{ttl})}\\ &+ \frac{(n^2+3n+14)n\kappa\varepsilon^2}{24} - \frac{n^2}{n+1}\varepsilon - \frac{n}{2(n+1)}X.\end{aligned}$$

For the found solution to be meaningful, the $(n+1)$th chunk must be nonempty, i.e., $\omega_{n+1,n+1} > 0$. This has two implications.

1. $X > \frac{(n+1)(n+2)}{2}\varepsilon$, which is true as long as $n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon} - 3\right)\right\rfloor$.
2. $W_{(\text{ttl})} - (n/(n+1))X - \frac{1}{2}n\varepsilon > 0$, which implies $W_{(\text{ttl})} > \frac{1}{2}n(n+1)\varepsilon$ because $X \geq W_{(\text{ttl})}$. This inequality on $W_{(\text{ttl})}$ is true as long as $n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\text{ttl})}/\varepsilon} - 1\right)\right\rfloor$.

Because $X > \frac{1}{2}(n+1)(n+2)\varepsilon$, we have

$$A_n + \omega_{n+1,n+1} + (n+1)\varepsilon \;\leq\; \frac{n}{n+1}X - \frac{n}{2}\varepsilon + \frac{1}{2(n+1)\kappa} - \frac{1}{4}(n+2)\varepsilon + (n+1)\varepsilon \;\leq\; X.$$

For both Case 1 and Case 2, if either condition

$$\left[n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8X/\varepsilon} - 3\right)\right\rfloor\right] \quad \text{or} \quad \left[n \leq \left\lfloor \frac{1}{2}\left(\sqrt{1+8W_{(\text{ttl})}/\varepsilon} - 1\right)\right\rfloor\right]$$

does not hold, then there is no optimal schedule with $(n+1)$ nonempty chunks. (We will come back later to the case where one of these conditions does not hold.) If both conditions hold, then Case 1 always has an optimal schedule, but Case 2 may not have one.

To complete the proof, we must verify that the optimal regimen always corresponds to Case 1 (as suggested by the theorem), never to Case 2 (whenever Case 2 defines a valid solution). We accomplish this by considering two cases, depending on the size $W_{(\text{ttl})}$ of the workload. We show that the expected work completed under the regimen of Case 1 is never less than under the regimen of Case 2.

Case A: $W_{(\text{ttl})} \geq \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon$.

Under this hypothesis, and under Case 1, the workload that is actually deployed has size

$$W_{(\text{dpl})} \;=\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,$$

so that, in expectation,

$$\begin{aligned}\mathcal{E}^{(1)}(W_{(\text{ttl})}, n+1) &= W_{(\text{dpl})} - \frac{n+1}{2n}W_{(\text{dpl})}^2\kappa - \frac{n+1}{2}W_{(\text{dpl})}\varepsilon\kappa \\ &\quad + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa \\ &= \frac{n+1}{2(n+2)}X - \frac{n+1}{2}\varepsilon + \frac{(n+1)(n+2)(n+3)}{24}\varepsilon^2\kappa.\end{aligned}$$

units of work are completed. Moreover, because

$$\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon \;\leq\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon,$$

the most favorable value for $\mathcal{E}^{(1)}(W_{(\text{ttl})}, n+1)$ under Case 2 lies within the range of values for the current case. Because the value of $\mathcal{E}^{(1)}(W_{(\text{ttl})}, n+1)$ is constant whenever

$$W_{(\text{ttl})} \;\geq\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,$$

we can reach the desired conclusion by just showing that this value is no smaller than: $\mathcal{E}^{(2)}\left(\frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon, n+1\right)$. Thus, we need only focus on the specific value

$$W_{(\text{ttl}-\text{lim})} \;=\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon$$

for $W_{(\text{ttl})}$. For this value, we have:

$$\begin{aligned}\mathcal{E}^{(2)}(W_{(\text{ttl}-\text{lim})}, n+1) &= -W_{(\text{ttl}-\text{lim})}^2\kappa + \left(\frac{n-2}{2}\kappa\varepsilon + \frac{2n+1}{n+1}\right)W_{(\text{ttl}-\text{lim})} \\ &\quad + \frac{(n^2+3n+14)n\kappa\varepsilon^2}{24} - \frac{n^2}{n+1}\varepsilon - \frac{n}{2(n+1)}X \\ &= \frac{2n^2+2n+1}{4(n+1)^2}X - \frac{2n^2+3n+2}{4(n+1)}\varepsilon \\ &\quad + \frac{(n+2)(2n^2+5n+6)}{48}\varepsilon^2\kappa.\end{aligned}$$

By explicit calculation, we finally see that

$$\begin{aligned}\mathcal{E}^{(1)}(W_{(\mathrm{ttl})},n+1)-&\mathcal{E}^{(2)}(W_{(\mathrm{ttl-lim})},n+1)\\ &= \frac{\left(4+(n^4+6n^3+13n^2+12n+4)\kappa^2\varepsilon^2\right)n}{16(n+1)^2(n+2)\kappa}\\ &\quad - \frac{(4n^2+12n+8)\kappa\varepsilon n}{16(n+1)^2(n+2)\kappa}\\ &= \frac{\left(4+(n+1)^2(n+2)^2\kappa^2\varepsilon^2-4(n+1)(n+2)\kappa\varepsilon\right)n}{16(n+1)^2(n+2)\kappa}\\ &= \frac{\left((n+1)(n+2)\kappa\varepsilon-2\right)^2 n}{16(n+1)^2(n+2)\kappa}\\ &\geq 0.\end{aligned}$$

Case B: $W_{(\mathrm{ttl})} \leq \dfrac{n+1}{n+2}X - \dfrac{n+1}{2}\varepsilon$.

In this case, the regimen of Case 1 deploys all $W_{(\mathrm{ttl})}$ units of work, thereby completing, in expectation,

$$\begin{aligned}&\mathcal{E}^{(1)}(W_{(\mathrm{ttl})},n+1)\\ &\qquad = W_{(\mathrm{ttl})} - \frac{n+1}{2n}W_{(\mathrm{ttl})}^2\kappa - \frac{n+1}{2}W_{(\mathrm{ttl})}\varepsilon\kappa + \frac{(n-1)n(n+1)}{24}\varepsilon^2\kappa.\end{aligned}$$

units of work. Moreover,

$$\frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon \;\leq\; \frac{2n+1}{2n+2}X + \frac{n-2}{4}\varepsilon,$$

so that the regimen of Case 2 also deploys all $W_{(\mathrm{ttl})}$ units of work, thereby completing, in expectation,

$$\begin{aligned}\mathcal{E}^{(2)}(W_{(\mathrm{ttl})},n+1) = &- W_{(\mathrm{ttl})}^2\kappa + \left(\frac{n-2}{2}\kappa\varepsilon + \frac{2n+1}{n+1}\right)W_{(\mathrm{ttl})}\\ &+ \frac{(n^2+3n+14)n\kappa\varepsilon^2}{24} - \frac{n^2}{n+1}\varepsilon - \frac{n}{2(n+1)}X.\end{aligned}$$

units of work.

Explicit calculation now shows that

$$\begin{aligned}\mathcal{E}^{(1)}(W_{(\mathrm{ttl})},n+1)-&\mathcal{E}^{(2)}(W_{(\mathrm{ttl})},n+1)\\ &= \frac{n}{2(n+1)}W_{(\mathrm{ttl})}^2\kappa - \frac{n}{n+1}(1+(n+1)\varepsilon\kappa)W_{(\mathrm{ttl})}\\ &\quad + \frac{n}{2(n+1)}(1+2n\kappa\varepsilon-(n+1)\kappa^2\varepsilon^2)X.\end{aligned}$$

Viewed as a function of $W_{(\mathrm{ttl})}$, this difference is, thus, unimodal, decreasing up to its global minimum, which occurs at $W_{(\mathrm{ttl})} = X + (n+1)\varepsilon$, and increasing thereafter. The largest value of $W_{(\mathrm{ttl})}$ allowed by the current case is

$$W_{(\mathrm{ttl-max})} \;=\; \frac{n+1}{n+2}X - \frac{n+1}{2}\varepsilon,$$

so this is also the value on which the difference $\mathcal{E}^{(1)}(W_{(\mathrm{ttl})},n+1) - \mathcal{E}^{(2)}(W_{(\mathrm{ttl})},n+1)$ reaches its minimum within its domain of validity. Thus, we need only focus on the behavior of the difference

at the value $W_{(\mathrm{ttl})} = W_{(\mathrm{ttl}-\max)}$. At this value,

$$\begin{aligned}\mathcal{E}^{(1)}(W_{(\mathrm{ttl}-\max)}, n+1) - \mathcal{E}^{(2)}(W_{(\mathrm{ttl}-\max)}, n+1) \\ = \frac{n(5n+1)\varepsilon^2\kappa}{8} + \frac{(n-1)n\varepsilon}{2(n+1)(n+2)} \\ + \frac{n}{2(n+1)(n+2)^2\kappa}.\end{aligned}$$

This quantity is obviously positive, which means that $\mathcal{E}^{(1)}(W_{(\mathrm{ttl}-\max)}, n+1) > \mathcal{E}^{(2)}(W_{(\mathrm{ttl}-\max)}, n+1)$.

We thus see that, for workloads of any size $W_{(\mathrm{ttl})}$, one completes at least as much expected work via the schedule of Case 1 as via the schedule of Case 2.

In summation, if

$$n \;\le\; \min\left\{\left\lfloor\frac{1}{2}\left(\sqrt{1+8X/\varepsilon}-3\right)\right\rfloor,\; \left\lfloor\frac{1}{2}\left(\sqrt{1+8W_{(\mathrm{ttl})}/\varepsilon}-1\right)\right\rfloor\right\}, \tag{13}$$

then Case 1 specifies the optimal schedule that uses no more than $n+1$ chunks. Of course, this inequality translates to the conditions of the theorem (where it is written for n chunks instead of $n+1$).

Note that if n exceeds either quantity in the minimization of (13), then one never improves the expected amount of work completed by deploying the workload in more than n chunks. This is another consequence of our remark about A_n at the beginning of this proof. If there exists a value of m for which there exists a schedule $\mathcal{S}$ that uses $\ge n+1$ nonempty chunks, then replacing the first $n+1$ chunks in this solution with the optimal solution for n chunks, using a workload equal to the first $n+1$ chunks of $\mathcal{S}$, yields a schedule that, in expectation, completes strictly more work than $\mathcal{S}$. □

4 Scheduling for Two Remote Computers

Before we approach the general case of p remote computers, we study the case of *two* remote computers, in order to adduce principles that will be useful in the general case. We first establish characteristics of optimal schedules under general risk functions, then restrict attention to the linear risk model. Throughout this section, we consider two remote computers, P_1 and P_2, under the *free-initiation* model.

4.1 Two Remote Computers under General Risk

Focus on a distribution of work to P_1 and P_2 under which, for $i = 1, 2$, P_i receives n_i chunks to execute, call them $\mathcal{W}_{i,1}, \ldots, \mathcal{W}_{i,n_i}$, to be scheduled in this order; as usual we denote $|\mathcal{W}_{i,j}|$ by $\omega_{i,j}$. We do not assume any *a priori* relation between the way P_1 and P_2 break their allocated work into chunks; in particular, any work that is allocated to both P_1 and P_2 may be chunked quite differently on the two machines.

Theorem 4. (Two remote computers: free-initiation model; general risk)
Let Σ be a schedule for two remote computers, P_1 and P_2. Say that, for both P_1 and P_2, the probability of being interrupted never decreases as a computer processes more work. There exists a schedule Σ' for P_1 and P_2 that, in expectation, completes as much work as does Σ and that satisfies the following three properties; cf. Fig. 1.

Maximal work deployment. *Σ' deploys as much of the overall workload as possible. Therefore, the workloads it deploys to P_1 and P_2 can overlap only if their union is the entire overall workload.*

Local work priority. Σ' *has* P_1 *(resp.,* P_2*) process all of the allocated work that it does* not *share with* P_2 *(resp.,* P_1*) before it processes any shared work.*

Shared work "mirroring." Σ' *has* P_1 *and* P_2 *process their shared work "in opposite orders." Specifically, say that* P_1 *chops its allocated work into chunks* $\mathcal{W}_{1,1}, \ldots, \mathcal{W}_{1,n_1}$*, while* P_2 *chops its allocated work into chunks* $\mathcal{W}_{2,1}, \ldots, \mathcal{W}_{2,n_2}$*.*

Say that there exist chunk-indices a_1*,* $b_1 > a_1$ *for* P_1*, and* a_2*,* $b_2 > a_2$ *for* P_2 *such that: chunks* $\mathcal{W}_{1,a_1}$ *and* $\mathcal{W}_{2,a_2}$ *both contain a shared "piece of work"* A*, and chunks* $\mathcal{W}_{1,b_1}$ *and* $\mathcal{W}_{2,b_2}$ *both contain a shared "piece of work"* B*.*

Then if Σ' *has* P_1 *execute* A *before* B *(i.e.,* P_1 *executes chunk* $\mathcal{W}_{1,a_1}$ *before chunk* $\mathcal{W}_{1,b_1}$*), then* Σ' *has* P_2 *execute* B *before* A *(i.e.,* P_2 *executes chunk* $\mathcal{W}_{2,b_2}$ *before chunk* $\mathcal{W}_{2,a_2}$*).*

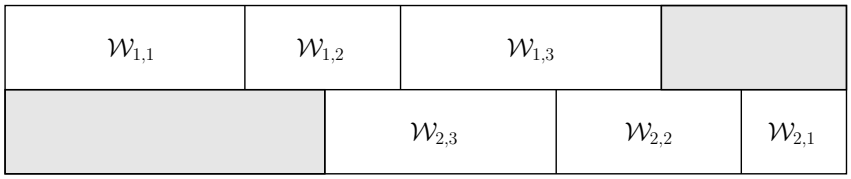


Figure 1: The shape of an optimal schedule for two computers, as described in Theorem 4; $n_1 = n_2 = 3$. The top row displays P_1's chunks, the bottom row P_2's. Vertically aligned parts of chunks correspond to shared work; shaded areas depict unallocated work (e.g., none of the work in $\mathcal{W}_{2,1}$ is allocated to P_1).

Proof. The strategy. We devise a cut-and-paste argument for each of the theorem's three characteristics in turn. Each time, we begin with an arbitrary schedule Σ that does not have that characteristic, and we show how to alter Σ to a schedule Σ' that does have the characteristic and that, in expectation, completes as much work as does Σ. In order to achieve the required alterations, we must refine our description of workloads. Specifically, we now describe the overall workload via a partition $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ of *pieces of work*, that has the following property. For $j = 1, 2$, each piece $\mathcal{X}_i \in \mathcal{X}$ is either included within a chunk of P_j, or it is disjoint from each chunk of P_j. We define, for $j = 1, 2$ and $i = 1, \ldots, m$, the indicator

$$\delta_j(i) \;=\; \begin{cases} 0 & \text{when } \mathcal{X}_i \in \mathcal{X} \text{ does not intersect} \\ & \text{any chunk of } P_j \\ 1 & \text{when } \mathcal{X}_i \text{ is contained within a chunk} \\ & \text{of } P_j, \text{ say, chunk } \sigma_j(i) \\ & \text{— so that } \mathcal{X}_i \subset \mathcal{W}_{j,\sigma_j(i)} \end{cases}$$

We now specify the expectation, $\mathcal{E}$, of $\mathsf{W}_{(\mathrm{cmp})}$ under this new specification of the deployment of chunks to P_1 and P_2. The probability that piece $\mathcal{X}_i \in \mathcal{X}$ is computed successfully is:

- 0 if $\mathcal{X}_i$ is not allocated to either P_1 or P_2;
- $1 - Pr\left(\sum_{j=1}^{\sigma_k(i)} \omega_{k,j}\right)$ if $\mathcal{X}_k$ is allocated only to P_k i.e., if $\delta_k(i)(1 - \delta_{k'}(i)) = 1$, where $\{k, k'\} = \{1, 2\}$;
- $1 - Pr\left(\sum_{j=1}^{\sigma_1(i)} \omega_{1,j}\right) Pr\left(\sum_{j=1}^{\sigma_2(i)} \omega_{2,j}\right)$ if $\mathcal{X}_i$ is allocated to both P_1 and P_2; i.e., if $\delta_1(i)\delta_2(i) = 1$.

We thus have

$$\mathcal{E} = \sum_{i=1}^{m} |\mathcal{X}_i| \cdot \Xi_i \tag{14}$$

where

$$\begin{aligned}\Xi_i &= \delta_1(i)\delta_2(i)\left(1 - Pr\left(\sum_{j=1}^{\sigma_1(i)}\omega_{1,j}\right)Pr\left(\sum_{j=1}^{\sigma_2(i)}\omega_{2,j}\right)\right)\\ &\quad + \delta_1(i)(1-\delta_2(i))\left(1 - Pr\left(\sum_{j=1}^{\sigma_1(i)}\omega_{1,j}\right)\right)\\ &\quad + (1-\delta_1(i))\delta_2(i)\left(1 - Pr\left(\sum_{j=1}^{\sigma_2(i)}\omega_{2,j}\right)\right).\end{aligned}$$

The alterations. We now look at each of our three characteristics in turn, performing the following process for each. We begin with a schedule $\Sigma^{(0)}$ that, in expectation, completes $E^{(0)}$ units of work. Say, for induction, that we now have a schedule $\Sigma^{(r)}$ that completes $E^{(r)}$ units of work. We describe how to alter $\Sigma^{(r)}$ to obtain a schedule $\Sigma^{(r+1)}$ that, in a sense, comes closer to enjoying the current characteristic and that, in expectation, completes $E^{(r+1)} \geq E^{(r)}$ units of work. We prove that a finite sequence of such alterations convert $\Sigma^{(0)}$ to a schedule Σ that enjoys the characteristic.

Maximal work deployment. Say that schedule Σ deploys some portion of the overall workload to both P_1 and P_2, while it leaves some other piece unallocated to either:

- the doubly allocated portion is a piece $\mathcal{X}_i \in \mathcal{X}$;
- the unallocated work is a piece $\mathcal{X}_j \in \mathcal{X}$.

To alter schedule Σ, we wish somehow to swap some doubly allocated work for an equal amount of unallocated work. This is easy when $|\mathcal{X}_i| = |\mathcal{X}_j|$. Otherwise, we achieve this goal as follows.

1. If $|\mathcal{X}_j| < |\mathcal{X}_i|$, then we invoke divisibility to subdivide $\mathcal{X}_i$ into one piece, $\mathcal{A}$, of size $|\mathcal{X}_j|$ and another of size $|\mathcal{X}_i| - |\mathcal{A}|$. We swap $\mathcal{B} = \mathcal{X}_j$ for $\mathcal{A}$ in the chunk of P_1 that contains $\mathcal{X}_i$.
2. If $|\mathcal{X}_j| > |\mathcal{X}_i|$, then we invoke divisibility to subdivide $\mathcal{X}_j$ into one piece, $\mathcal{B}$, of size $|\mathcal{X}_i|$ and another of size $|\mathcal{X}_j| - |\mathcal{B}|$. We swap $\mathcal{B}$ for $\mathcal{A} = \mathcal{X}_i$ in the relevant chunk of P_1.

In case 1, $\Sigma^{(r+1)}$ has no more unallocated work and the maximal deployment rule is in force. In case 2, $\Sigma^{(r+1)}$ has one fewer piece of doubly allocated work. It follows that a finite sequence of alterations convert $\Sigma^{(0)}$ into a schedule that practices maximal work deployment. *We henceforth assume that Σ practices maximal work deployment.*

Local-work prioritization. Assume that under optimal schedule Σ, there exist $\mathcal{X}_i, \mathcal{X}_j \in \mathcal{X}$ such that:

1. Σ allocates $\mathcal{X}_i$ to P_1 but not to P_2; i.e., $\delta_1(i)(1-\delta_2(i)) = 1$;
2. Σ allocates $\mathcal{X}_j$ to both P_1 and P_2; i.e., $\delta_1(i)\delta_2(i) = 1$;
3. Σ attempts to execute $\mathcal{X}_j$ before $\mathcal{X}_i$ on P_1: symbolically, $\sigma_1(i) \geq \sigma_1(j)$.

We alter Σ to obtain a new schedule that comes closer to prioritizing local work. And, we do so in a way that (a) at least matches Σ's expected work production and (b) guarantees that a finite sequence of alterations produce a schedule that practices local work prioritization. We proceed as follows.

We codify the set of violations of local work prioritization via the set $\mathcal{V}$ that identifies every triplet of chunks that violate local work prioritization.

$$\mathcal{V} = \left\{ (k,k',l) \;\middle|\; \begin{array}{l} \mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'} \neq \emptyset \text{ (replicated work)} \\ \mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'} \neq \emptyset \text{ (local work)} \\ k < l \text{ (replicated starts before local)} \\ k \in [1,n_1], k' \in [1,n_2], l \in [1,n_1] \end{array} \right\}$$

To choose the alteration to apply to Σ at this step, we take any triplet $(k,k',l) \in \mathcal{V}$ whose first component, k, is minimal among all the first components of elements of $\mathcal{V}$, and whose third component, l, is maximal among all the third components of elements of $\mathcal{V}$ (one verifies easily that such an element always exists). From the perspective of P_1, we thus focus on the earliest-scheduled replicated chunk that is scheduled before the latest-scheduled unreplicated chunk.

(1) Say first that $|\mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}| \leq |\mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}|$. In this case, we alter Σ by swapping the piece of work $\mathcal{A} = \mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}$ from $\mathcal{W}_{1,l}$ with an arbitrarily chosen like-sized subset $\mathcal{B}$ of $\mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}$ in $\mathcal{W}_{1,k}$. After the swap, $\mathcal{V}$ no longer contains any element of the form (α, β, l), because chunk $\mathcal{W}_{1,l}$ now contains only replicated work. Furthermore, by choice of k, chunks $\mathcal{W}_{1,1}$ through $\mathcal{W}_{1,k-1}$ contain only work for P_1 that is not replicated on P_2. Thus, the swap reduces the number of violations.

(2) Say alternatively that $|\mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}| > |\mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}|$. In this case, we alter Σ by swapping the piece of work $\mathcal{B} = \mathcal{W}_{1,k} \cap \mathcal{W}_{2,k'}$ from $\mathcal{W}_{1,k}$ with an arbitrarily chosen like-sized subset $\mathcal{A}$ of $\mathcal{W}_{1,l} \setminus \bigcup_{l'=1}^{n_2} \mathcal{W}_{2,l'}$ in $\mathcal{W}_{1,l}$. After the swap, $\mathcal{V}$ no longer contains any element of the form (k, k', α). Furthermore, by definition of k, chunks $\mathcal{W}_{1,1}$ through $\mathcal{W}_{1,k-1}$ contain only work for P_1 that is not replicated on P_2. Thus, this swap also reduces the number of violations.

Clearly, at most $|\mathcal{V}|$ alterations are needed to convert Σ to a schedule that practices local-work prioritization *on computer* P_1. Because each alteration affects only the scheduling on P_1, we can now apply an analogous sequence of alterations that focus on violations by computer P_2. After this second round of alterations, we have finally converted Σ to a schedule that practices local work prioritization on *both* computers. *We henceforth assume that* Σ *practices local-work prioritization.*

"Mirroring" of replicated work. Say that, under schedule Σ, there are two partition elements, $\mathcal{X}_i$ and $\mathcal{X}_j$, such that:

1. Σ allocates both $\mathcal{X}_i$ and $\mathcal{X}_j$ to both P_1 and P_2; i.e., $\delta_1(i)\delta_2(i) = \delta_1(j)\delta_2(j) = 1$;
2. Σ attempts to execute $\mathcal{X}_i$ after $\mathcal{X}_j$ on both P_1 and P_2: symbolically, $[\sigma_1(i) \geq \sigma_1(j)]$ and $[\sigma_2(i) \geq \sigma_2(j)]$.

We craft a sequence of alterations to Σ that produce a schedule that practices the mirroring of replicated work. Essentially, at each step, we identify a pair of pieces of work, $\mathcal{A}$ and $\mathcal{B}$, that violate mirroring in the way just described. We then swap $\mathcal{B}$ for $\mathcal{A}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$ and swap $\mathcal{A}$ for $\mathcal{B}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{B})}$, while leaving the schedule of P_2 unchanged.

How do we select the pieces to focus on at this step? Our job is somewhat simplified by our ability to focus entirely on replicated pieces of work—because of our assumption that Σ practices both maximal work deployment and local-work prioritization. We employ the following inductive process to choose the pieces from among pieces of replicated work that violate mirroring. Say, for induction, that we have times—so that the k pieces of replicated work that P_1 is scheduled to (attempt to) execute *first* are the k pieces of replicated work that P_2 is scheduled to (attempt to) execute *last*, and that these pieces are executed in reverse orders on P_1 and P_2. We now select the $(k+1)$th piece of replicated work that P_1 is scheduled to (attempt to) execute, call it $\mathcal{X}_i$, and the kth from last piece of replicated work that P_2 is scheduled to (attempt to) execute, call it $\mathcal{X}_j$.

(1) If $\mathcal{X}_i = \mathcal{X}_j$, then there is no violation to undo.

(2) If $\mathcal{X}_i \neq \mathcal{X}_j$, then we select the pieces, $\mathcal{A}$ and $\mathcal{B}$, to swap in the $(k+1)$th alteration of Σ, in the following manner. (*a*) If $|\mathcal{X}_j| \geq |\mathcal{X}_i|$, then we have $\mathcal{X}_i$ play the role of $\mathcal{A}$, and we select as $\mathcal{B}$ any size-$|\mathcal{A}|$ subset of $\mathcal{X}_j$. After the swap, $\mathcal{B}$ is scheduled as the $(k+1)$th piece of replicated work for P_1 to (attempt to) execute and (in deference to mirroring) as the kth from last piece of replicated work for P_2 to (attempt to) execute. None of the first k pieces of replicated work for P_1 nor any of the last k pieces of replicated work for P_2 were affected by the swap. To conclude that the inductive process eventually terminates (successfully!) we first consider the number of chunks of P_1 (respectively P_2) that include only work from the first (resp. last) k pieces of replicated work. As the alterations progress, the numbers of such pieces never decrease. If an alteration does not increase either of these numbers, then we focus on the set of early chunks of P_2 that contain work that is replicated in the chunk of P_1 that we are working with:

$$\mathcal{V}_2 = \left\{ l < \sigma_2(\mathcal{X}_j) \mid \mathcal{W}_{1,\sigma_1(\mathcal{X}_i)} \cap \mathcal{W}_{2,l} \neq \emptyset \right\},$$

and the symmetrical set of chunks for P_1:

$$\mathcal{V}_1 = \left\{ l > \sigma_1(\mathcal{X}_i) \mid \mathcal{W}_{1,l} \cap \mathcal{W}_{2,\sigma_2(\mathcal{X}_j)} \neq \emptyset \right\}.$$

If we assume—as we may with no loss of generality—that we are working with a partition made of *maximal* elements, then the alteration decreases the set $\mathcal{V}_2$ by one element (namely, $\mathcal{W}_{2,\sigma_2(\mathcal{X}_i)}$)

and does not modify the set $\mathcal{V}_1$. (b) The case when $|\mathcal{X}_{j'}| < |\mathcal{X}_{i'}|$ is symmetric with case (a), hence is left to the reader. We note only that, in that case, the alteration does not modify the set $\mathcal{V}_2$, and it decreases the set $\mathcal{V}_1$ by one element (namely, $\mathcal{W}_{1,\sigma_1(\mathcal{X}_j)}$)

Validating the alterations. To complete the proof, we need only verify that each of the schedule alterations we have described cannot produce a schedule that completes less work than schedule Σ does. Rather similar arguments verify this work preservation for each of the three characteristic. An important feature of our alterations that greatly simplifies these verifications is that, in each case, the relevant alteration affects only the terms in expression (14) that mention the pieces involved in the alteration.

Maximal work deployment. Recall that our alteration of schedule Σ in this case substituted piece $\mathcal{B}$ for piece $\mathcal{A}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$. Now, before this substitution, the total contribution to the expectation $\mathcal{E}$ of these pieces was:

$$|\mathcal{A}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \cdot 0.$$

After the substitution, this contribution becomes:

$$|\mathcal{A}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right).$$

Because $|\mathcal{A}| = |\mathcal{B}|$, the latter contribution is never less than the former, differing from it by the quantity

$$|\mathcal{A}| \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right) \cdot \left(1 - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right),$$

whose nonnegativity implies that the altered schedule completes at least as much work, in expectation, as does schedule Σ.

Local work prioritization. Recall that our alteration of schedule Σ in this case substituted a piece of local work $\mathcal{A}$ from $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$ with a piece of replicated work $\mathcal{B}$ of $\mathcal{W}_{1,\sigma_1(\mathcal{B})}$. Now, before this substitution, the total contribution to the expectation $\mathcal{E}$ of these pieces was:

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

After the substitution, the contribution becomes

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

Because $|\mathcal{A}| = |\mathcal{B}|$, we see that the substitution increases the overall expectation by the quantity

$$|\mathcal{A}| \times \left(Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right)\right) \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right),$$

This quantity is nonnegative because

- the probability $Pr(w)$ is nondecreasing in w;
- $\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k} \geq \sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}$, because $\sigma_1(\mathcal{A}) \geq \sigma_1(\mathcal{B})$.

The altered schedule completes, in expectation, at least as much work as Σ.

Shared work "mirroring." Recall that our alteration of schedule Σ in this case swapped $\mathcal{B}$ for $\mathcal{A}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{A})}$ and swapped $\mathcal{A}$ for $\mathcal{B}$ in chunk $\mathcal{W}_{1,\sigma_1(\mathcal{B})}$, while leaving the schedule of P_2 unchanged. Now, before this substitution, the total contribution to the expectation $\mathcal{E}$ of these pieces was:

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

After the substitution, their contribution becomes:

$$|\mathcal{A}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right) + |\mathcal{B}| \times \left(1 - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right) Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right)\right).$$

Because $|\mathcal{A}| = |\mathcal{B}|$, the substitutions have increased the overall expectation by the quantity:

$$|\mathcal{A}| \times \left(Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{B})} \omega_{1,k}\right) - Pr\left(\sum_{k=1}^{\sigma_1(\mathcal{A})} \omega_{1,k}\right)\right)\left(Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{B})} \omega_{2,k}\right) - Pr\left(\sum_{k=1}^{\sigma_2(\mathcal{A})} \omega_{2,k}\right)\right).$$

This quantity is nonnegative because $\mathcal{A}$ and $\mathcal{B}$ are processed in the same order on P_1 and on P_2. The altered schedule thus completes, in expectation, at least as much work as Σ. □

4.2 Two Remote Computers under Linear Risk

4.2.1 Allocating work in a single chunk

We now specialize from general risk functions to the linear risk function. We first consider the case wherein each computer processes its allocated work as a single chunk. Even this simple case turns out to be surprisingly difficult to schedule optimally when there is more than one remote computer. Indeed, our experience with this case has led us to abandon the quest for exactly optimal schedules, in favor of the more easily accessed *asymptotically* optimal schedules.

> "Asymptotically optimal" here means that the expected amount of work completed by these schedules deviates from exact maximality by an amount that shrinks as the size of the workload grows without bound.

To render the single-chunk scheduling problem tractable, we restrict attention to schedules that are *symmetric,* in the sense that they allocate the same amount of work to each remote computer. It seems intuitive that there is always a symmetric schedule among the optimal single-chunk schedules, but we have yet to verify this.

Say that our workload consists of $W_{(\text{ttl})}$ units of work that we somehow order linearly. We denote by $\langle a, b\rangle$ the sub-workload obtained by eliminating: the initial a units of work and all work beyond the initial b units. For instance, $\langle 0, W_{(\text{ttl})}\rangle$ denotes the entire workload, $\langle 0, \frac{1}{2}W_{(\text{ttl})}\rangle$ denotes the first half of the workload, and $\langle \frac{1}{2}W_{(\text{ttl})}, W_{(\text{ttl})}\rangle$ denotes the last half of the workload.

Theorem 5. (Two remote computers: linear risk; single chunk allocation)
Say that we wish to deploy $W_{(\text{ttl})}$ units of work on two computers, deploying a single chunk per computer. The following symmetric schedule Σ completes, in expectation, a maximum amount of work.

- *If $W_{(\mathrm{ttl})} \leq \frac{1}{2}X$, then Σ deploys the entire workload on both remote computers; symbolically, $\mathcal{W}_{1,1} = \mathcal{W}_{2,1} = \langle 0, W_{(\mathrm{ttl})} \rangle$;*
- *if $\frac{1}{2}X < W_{(\mathrm{ttl})} \leq X$, then Σ deploys the first half of the workload on one computer and the second half on the other; symbolically, $\mathcal{W}_{1,1} = \langle 0, \frac{1}{2}W_{(\mathrm{ttl})} \rangle$, and $\mathcal{W}_{2,1} = \langle \frac{1}{2}W_{(\mathrm{ttl})}, W_{(\mathrm{ttl})} \rangle$;*
- *if $X < W_{(\mathrm{ttl})}$, then Σ deploys only X units of the workload, allocating the first half to one computer and the second half to the other; symbolically, $\mathcal{W}_{1,1} = \langle 0, \frac{1}{2}X \rangle$, and $\mathcal{W}_{2,1} = \langle \frac{1}{2}X, X \rangle$.*

Proof. Our derivation of the optimal schedule builds on the following principle (which we have encountered before). When we deploy work as a single chunk, we never make that chunk as large as X, for then we risk certain interruption, hence, in expectation, completes no work. In order to see how to optimally deploy work as a single chunk, we consider separately schedules that allow overlapping deployments to the two computers and those that do not.

Disjoint allocations. Focus first on schedules that deploy non-overlapping workloads to the two remote computers. These two workloads, $\mathcal{W}_{1,1}$ and $\mathcal{W}_{2,1}$, are independent, so we can invoke Theorem 2 to discover their optimal sizes. We see that the optimal strategy is to deploy $W_{(\mathrm{dpl})} = \min\{W_{(\mathrm{ttl})}, X\}$ units of work in total. We determine the optimal allocation of this work to the remote computers, in respective chunk sizes $\omega_{1,1}$ and $\omega_{2,1} = Z - \omega_{1,1}$, by considering the expectation of $\mathsf{W}_{(\mathrm{cmp})}$.

$$\begin{aligned} \mathcal{E} &= \omega_{1,1}\left((1-\omega_{1,1}\kappa) + (W_{(\mathrm{dpl})} - \omega_{1,1})(1-(W_{(\mathrm{dpl})} - \omega_{1,1})\kappa)\right) \\ &= -2\omega_{1,1}^2\kappa + 2\omega_{1,1}Z\kappa + Z - Z^2\kappa. \end{aligned}$$

Easily, $\mathcal{E}$ is maximized when $\omega_{1,1} = \omega_{2,1} = \frac{1}{2}W_{(\mathrm{dpl})}$. The optimal value of $\mathcal{E}$ is, then, $\mathcal{E} = W_{(\mathrm{ttl})} - \frac{1}{2}W_{(\mathrm{dpl})}^2\kappa$. (Note that we did not need to *assume* that the optimal schedule is symmetric in this case; we actually proved that it should be.)

Overlapping allocations to the two computers. The principle enunciated at the beginning of this proof implies that an optimal schedule that deploys overlapping workloads to the two remote computers never allocates a full X units of work to either computer. We can, therefore, simplify our calculations by restricting attention henceforth to the case $W_{(\mathrm{ttl})} < 2X$. Since we consider only symmetric schedules, the common size s of the allocations to both computers satisfies $s \geq \frac{W_{(\mathrm{ttl})}}{2}$, by Theorem 4. We thus obtain the following expression for the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ as a function of s.

$$\begin{aligned} \mathcal{E}(s) &= 2(W_{(\mathrm{ttl})} - s)(1 - s\kappa) + (2s - W_{(\mathrm{ttl})})(1 - s^2\kappa^2) \\ &= -2s^3\kappa^2 + (2 + W_{(\mathrm{ttl})}\kappa)s^2\kappa - 2sW_{(\mathrm{ttl})}\kappa + W_{(\mathrm{ttl})}. \end{aligned}$$

We seek the maximizing value of s.

$$\mathcal{E}'(s) = \frac{\mathrm{d}}{\mathrm{d}s}\mathcal{E}(s) = 2[-3s^2\kappa + (2 + W_{(\mathrm{ttl})}\kappa)s - W_{(\mathrm{ttl})}]\kappa.$$

The discriminant of the bracketed quadratic polynomial is

$$\Delta = W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4 = \left(W_{(\mathrm{ttl})}\kappa - 2\left(2+\sqrt{3}\right)\right)\left(W_{(\mathrm{ttl})}\kappa - 2\left(2-\sqrt{3}\right)\right).$$

Because $W_{(\mathrm{ttl})} < 2X$ we have, $W_{(\mathrm{ttl})}\kappa < 2(2+\sqrt{3})$. We branch on the relative sizes of $W_{(\mathrm{ttl})}$ and $2(2-\sqrt{3})X$:

$\underline{W_{(\mathrm{ttl})} > 2(2-\sqrt{3})X}$. In this case, $\Delta < 0$, so the polynomial has no real roots, and $\mathcal{E}(s)$ is decreasing with s. Because $s \in [\frac{1}{2}W_{(\mathrm{ttl})}, W_{(\mathrm{ttl})}]$, $\mathcal{E}(s)$ is maximized when $s = W_{(\mathrm{ttl})}/2$.

$\underline{W_{(\mathrm{ttl})} \leq 2(2-\sqrt{3})X}$. This case is far more complicated than its predecessor. Let us denote the two roots of our quadratic polynomial by s_- and s_+, as follows:

$$s_- = \frac{2 + W_{(\mathrm{ttl})}\kappa - \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa} \quad \text{and}$$

$$s_+ \;=\; \frac{2 + W_{(\mathrm{ttl})}\kappa + \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa}.$$

One sees that $\mathcal{E}(s)$ decreases as s progresses from $-\infty$ to s_-, then increases as s progresses from s_- to s_+, then decreases once more as s increases beyond s_+. We must determine how these three intervals overlap $\mathcal{E}$'s domain of validity, viz., $s \in [\frac{1}{2}W_{(\mathrm{ttl})},\ W_{(\mathrm{ttl})}]$.

We note first that $\frac{1}{2}W_{(\mathrm{ttl})} \le s_-$. Indeed:

$$\begin{array}{ll} & W_{(\mathrm{ttl})}/2 \ge s_- \\ \text{just when} & W_{(\mathrm{ttl})}/2 \ge \dfrac{2 + W_{(\mathrm{ttl})}\kappa - \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa} \\ \\ \text{just when} & \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4} \ge 2(1 - W_{(\mathrm{ttl})}\kappa) \\ \text{only if} & 0 \ge 3W_{(\mathrm{ttl})}^2\kappa^2. \end{array}$$

We invoke here the fact that $W_{(\mathrm{ttl})}\kappa \le 1$, because $W_{(\mathrm{ttl})} \le 2(2-\sqrt{3})X \le X$. We remark next that $\mathcal{E}'(W_{(\mathrm{ttl})}) = 2W_{(\mathrm{ttl})}\kappa(1 - 2W_{(\mathrm{ttl})}\kappa)$, so that $\mathcal{E}'(W_{(\mathrm{ttl})}) \ge 0$ and $W_{(\mathrm{ttl})} \in [s_-, s_+]$ when $W_{(\mathrm{ttl})} \le \frac{X}{2}$; moreover, $W_{(\mathrm{ttl})} > s_+$ when $W_{(\mathrm{ttl})} > \frac{1}{2}X$. Indeed, if we assume that $\frac{1}{2}X < W_{(\mathrm{ttl})} \le s_+$ (the lower bound implying $5W_{(\mathrm{ttl})}\kappa - 2 \ge 0$), then we reach a contradiction:

$$\begin{array}{lllll} & W_{(\mathrm{ttl})} & \le & s_+ \\ \text{just when} & W_{(\mathrm{ttl})} & \le & \dfrac{2 + W_{(\mathrm{ttl})}\kappa + \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4}}{6\kappa} \\ \\ \text{just when} & 5W_{(\mathrm{ttl})}\kappa - 2 & \le & \sqrt{W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4} \\ \text{only if} & 2W_{(\mathrm{ttl})}\kappa & \le & 1. \end{array}$$

So, once again we have two cases to consider:

$\underline{W_{(\mathrm{ttl})} \le X/2}$. In this case, we have $W_{(\mathrm{ttl})} \in [s_-, s_+]$, so that $\mathcal{E}(s)$ achieves its maximum either when $s = \frac{1}{2}W_{(\mathrm{ttl})}$ or when $s = W_{(\mathrm{ttl})}$. Hence $\mathcal{E}(s)$'s maximum is either

$$\mathcal{E}(W_{(\mathrm{ttl})}/2) \;=\; W_{(\mathrm{ttl})} - \frac{1}{2}W_{(\mathrm{ttl})}^2\kappa \qquad \text{or} \qquad \mathcal{E}(W_{(\mathrm{ttl})}) \;=\; W_{(\mathrm{ttl})} - W_{(\mathrm{ttl})}^3\kappa^2.$$

When $W_{(\mathrm{ttl})} \le \frac{1}{2}X$, which is the case here, the latter value dominates, so the optimal deployment is $\omega_{1,1} = \omega_{2,1} = W_{(\mathrm{ttl})}$.

$\underline{W_{(\mathrm{ttl})} > X/2}$. In this case, $W_{(\mathrm{ttl})} > s_+$, so that $\mathcal{E}(s)$ achieves its maximum either when $s = W_{(\mathrm{ttl})}/2$ or when $s = s_+$. We compare the values at these points by computing both $\mathcal{E}(s_+)$ and $\mathcal{E}(s_+) - \mathcal{E}(W_{(\mathrm{ttl})}/2)$. We find that

$$\mathcal{E}(s_+) \;=\; \frac{(W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4)^{\frac{3}{2}} + W_{(\mathrm{ttl})}^3\kappa^3 - 12W_{(\mathrm{ttl})}^2\kappa^2 + 30W_{(\mathrm{ttl})}\kappa + 8}{54\kappa}$$

and

$$\mathcal{E}(s_+) - \mathcal{E}(W_{(\mathrm{ttl})}/2) \;= \\ \frac{[W_{(\mathrm{ttl})}^2\kappa^2 - 8W_{(\mathrm{ttl})}\kappa + 4]^{3/2} + [W_{(\mathrm{ttl})}^3\kappa^3 + 15W_{(\mathrm{ttl})}^2\kappa^2 - 24W_{(\mathrm{ttl})}\kappa + 8]}{54\kappa}. \quad (15)$$

Easily, both of the bracketed polynomials in (15) *decrease* as $W_{(\mathrm{ttl})}$ progresses along its current hypothesized interval, from $\frac{1}{2}X$ through $2(2-\sqrt{3})X$; therefore, the difference $\mathcal{E}(s_+) - \mathcal{E}(W_{(\mathrm{ttl})}/2)$ decreases as $W_{(\mathrm{ttl})}$ proceeds along the same interval. Since the difference vanishes at the point $W_{(\mathrm{ttl})} = \frac{1}{2}X$, we conclude that the optimal deployment in this case is $\omega_{1,1} = \omega_{2,1} = \frac{1}{2}W_{(\mathrm{ttl})}$. $\square$

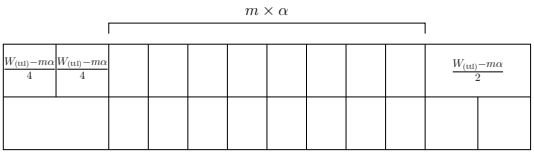


Figure 2: Counterexample to the optimality of schedules that employ equal-size chunks.

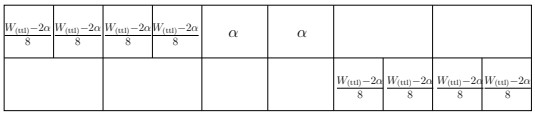


Figure 3: Counterexample to the optimality of the schedule of Fig. 2.

Moving beyond Theorem 5. The preceding analysis determines the optimal schedule for only two remote computers that each process their allocations as single chunks. The complexity of even this simple case has led us to abandon our focus on *exactly* optimal schedules in the sequel, in favor of a hopefully more tractable search for schedules that are *asymptotically* optimal.

> Since one can view schedules under the free-initiation model as "asymptotic versions" of schedules under the charged-initiation model —cf. Theorem 1—our shift in focus is not a drastic one.

This shift in focus notwithstanding, it is worth seeking significant restricted situations wherein one can tractably discover exactly optimal schedules. One obvious candidate for special consideration is the family of schedules that allocate the entire workload to each remote computer—which seems to be desirable when $W_{(\mathrm{ttl})}\kappa$ is small enough. We conjecture that, for such schedules, an optimal strategy would have the two computers chop the workload into chunks of the same size and then process these chunks in "opposite orders" (as defined in the third property of Theorem 4). When all remote computers chop the workload into n chunks, this scheduling strategy completes, in expectation,

$$\mathcal{E} = W_{(\mathrm{ttl})} - \frac{W_{(\mathrm{ttl})}^3\kappa^2}{6}\left(1 + \frac{3}{n} + \frac{2}{n^2}\right)$$

units of work (cf. Theorem 6 below). Extensive numerical simulations suggest that such a scheduling strategy is, indeed, optimal as long as $n \leq 3$. However, we know that the strategy is *suboptimal* once one allows n to exceed 3. Indeed, for $n = 4$, the strategy completes, in expectation, $W_{(\mathrm{ttl})} - \frac{5}{16}W_{(\mathrm{ttl})}^3\kappa^2$ units of work, which is strictly less than the $W_{(\mathrm{ttl})} - \frac{757-73\sqrt{73}}{432}W_{(\mathrm{ttl})}^3\kappa^2$ units completed, in expectation, by the strategy specified schematically in Fig. 2 (with $m = 1$ and $\alpha = \frac{\sqrt{73}-7}{6}W_{(\mathrm{ttl})}$).

> The boxes in Figs. 2 and 3 contain chunk sizes. In Fig. 2, for instance, each computer uses m chunks of size α, two chunks of size $\frac{1}{4}(W - m\alpha)$, and one chunk of size $\frac{1}{2}(W - m\alpha)$.

Furthermore, the schedule in Fig. 2 is suboptimal as soon as we allow computers to chop work into eight chunks. To wit, Fig. 3 presents an 8-chunk schedule that completes, in expectation, $W_{(\mathrm{ttl})} - \frac{229-44\sqrt{22}}{98}W_{(\mathrm{ttl})}^3\kappa^2 \approx W_{(\mathrm{ttl})} - 0.230834W_{(\mathrm{ttl})}^3\kappa^2$ units of work, when $\alpha = \frac{4\sqrt{22}-17}{14}W_{(\mathrm{ttl})}$, while the schedule of Fig. 2, using 8 chunks per computer (specifically, $m = 5$ and $\alpha = \frac{19-\sqrt{193}}{42}W_{(\mathrm{ttl})}$) completes, in expectation, $W_{(\mathrm{ttl})} - \frac{18293-965\sqrt{193}}{21168}W_{(\mathrm{ttl})}^3\kappa^2 \approx W_{(\mathrm{ttl})} - 0.230857W_{(\mathrm{ttl})}^3\kappa^2$ units of work. (The schedule of Fig. 3 is not even optimal for 8 chunks, but the best schedule we found numerically was almost identical but slightly less regular.)

The increasing complexities of the preceding "counterexample" schedules suggest how hard it will be to search for, and characterize, exactly optimal schedules—even in the presence of simplifying assumptions, such as that the whole workload is distributed to each computer. Since our simulations suggest that simple regular solutions often complete, in expectation, almost as much work as do complex exactly optimal schedules, we henceforth aim for simply structured asymptotically optimal schedules.

4.2.2 Asymptotically optimal schedules

This section is devoted to Algorithm 1, whose prescribed schedules for two remote computers branch on the value of $W_{(\mathrm{ttl})}\kappa$. We show in Theorem 6 that the proposed schedules are all asymptotically optimal; they are exactly optimal when $W_{(\mathrm{ttl})}\kappa \geq 2$.

Algorithm 1: Scheduling for 2 computers using at most n chunks per computer

if $W_{(\mathrm{ttl})} \geq 2X$ **then**

  $\forall i \in [1,n],\ \mathcal{W}_{1,i} \leftarrow \left\langle \frac{i-1}{n} \cdot \frac{n}{n+1}X,\ \frac{i}{n} \cdot \frac{n}{n+1}X \right\rangle$

  $\forall i \in [1,n],\ \mathcal{W}_{2,i} \leftarrow \left\langle W_{(\mathrm{ttl})} - \frac{i}{n} \cdot \frac{n}{n+1}X,\ W_{(\mathrm{ttl})} - \frac{i-1}{n} \cdot \frac{n}{n+1}X \right\rangle$

if $W_{(\mathrm{ttl})} \leq X$ **then**

  $\forall i \in [1,n],\ \mathcal{W}_{1,i} = \mathcal{W}_{2,n-i+1} \leftarrow \left\langle \frac{i-1}{n}W_{(\mathrm{ttl})},\ \frac{i}{n}W_{(\mathrm{ttl})} \right\rangle$

if $X < W_{(\mathrm{ttl})} < 2X$ **then**

  $\ell \leftarrow \lfloor n/3 \rfloor$

  $\forall i \in [1,\ell],\ \mathcal{W}_{1,i} \leftarrow \left\langle \frac{i-1}{\ell}(W_{(\mathrm{ttl})} - X),\ \frac{i}{\ell}(W_{(\mathrm{ttl})} - X) \right\rangle$

  $\forall i \in [1,\ell],\ \mathcal{W}_{2,i} \leftarrow \left\langle W_{(\mathrm{ttl})} - \frac{i}{\ell}(W_{(\mathrm{ttl})} - X),\ W_{(\mathrm{ttl})} - \frac{i-1}{\ell}(W_{(\mathrm{ttl})} - X) \right\rangle$

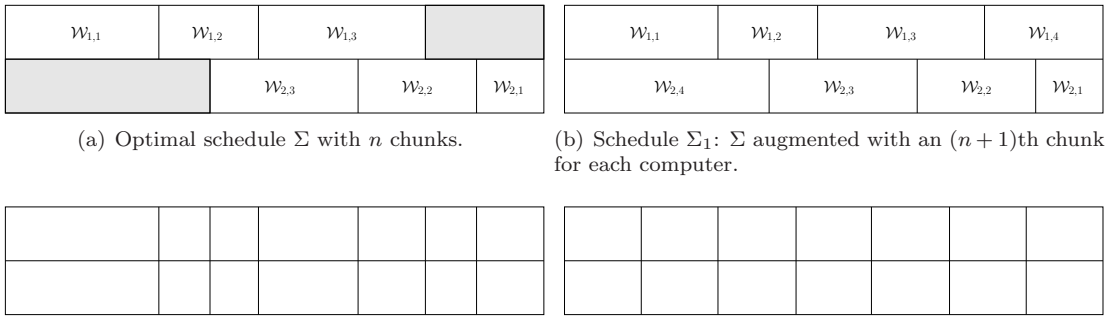
  $\forall i \in [1,2l],$

  $\mathcal{W}_{1,l+i} = \mathcal{W}_{2,3l-i+1} \leftarrow \left\langle (W_{(\mathrm{ttl})} - X) + \frac{i-1}{2\ell}(2X - W_{(\mathrm{ttl})}),\ (W_{(\mathrm{ttl})} - X) + \frac{i}{2\ell}(2X - W_{(\mathrm{ttl})}) \right\rangle$

Theorem 6. *The schedules specified by Algorithm 1 are:*

1. *optimal when $W_{(\mathrm{ttl})} \geq 2X$;*
   In expectation, the schedules complete
   $$E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \textit{Algorithm 1}(n)) = \frac{n-1}{n}X$$
   units of work, which tends to[9] *X;*
2. *asymptotically optimal when $W_{(\mathrm{ttl})} \leq X$;*
   In expectation, the schedules complete
   $$E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \textit{Algorithm 1}(n)) = W_{(\mathrm{ttl})} - \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2\left(1 + \frac{3}{n} + \frac{2}{n^2}\right)$$
   units of work, which tends to $W_{(\mathrm{ttl})} - \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2$;
3. *asymptotically optimal when $X < W_{(\mathrm{ttl})} < 2X$.*
   Letting $\ell = \lfloor n/3 \rfloor$, in expectation, the schedules complete
   $$\begin{aligned} E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \textit{Algorithm 1}(n)) \ &= \ 2W_{(\mathrm{ttl})} - \frac{1}{3}X - W_{(\mathrm{ttl})}^2\kappa + \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2 \\ &\quad + \frac{1}{\ell}\left(\left(1 + \frac{1}{\ell}\right)W_{(\mathrm{ttl})} - \left(1 + \frac{2}{3\ell}\right)X - \frac{1}{2\ell}W_{(\mathrm{ttl})}^2\kappa - \frac{1}{4}\left(1 - \frac{1}{3\ell}\right)W_{(\mathrm{ttl})}^3\kappa^2\right) \end{aligned}$$
   units of work, which tends to $2W_{(\mathrm{ttl})} - \frac{1}{3}X - W_{(\mathrm{ttl})}^2\kappa + \frac{1}{6}W_{(\mathrm{ttl})}^3\kappa^2$.

[9] "tends to" means "as n grows without bound."



(a) Optimal schedule Σ with n chunks.

(b) Schedule Σ_1: Σ augmented with an $(n+1)$th chunk for each computer.

(c) Schedule Σ_2: Σ_1 with chunks subdivided so that chunk boundaries coincide.

(d) Schedule Σ_3: Σ_2 via Algorithm 1 with $2n+1$ chunks.

Figure 4: Schedule transformations that prove the asymptotic optimality of Algorithm 1 when $W_{(\text{ttl})} \leq X$.

Proof. We prove the theorem's three assertions in turn.

1. Case: $W_{(\text{ttl})} \geq 2X$.
By definition, a computer is certain to be interrupted when processing a workload of size $\geq X$. Therefore, when $W_{(\text{ttl})} \geq 2X$, Theorem 4 tells us that the two remote computers are working on disjoint subsets of the workload. In this case, then, Theorem 2: (a) defines the sizes of these workloads and the way they are partitioned into chunks; (b) gives us the expectation of $\mathsf{W}_{(\text{cmp})}$.

2. Case: $W_{(\text{ttl})} \leq X$. We must prove that the proposed schedule is asymptotically optimal, and we must evaluate the resulting expected work production. Focus on an arbitrary positive integer n. Clearly, the expected work produced by Algorithm 1 when each remote computer's workload is partitioned into n chunks cannot exceed the analogous quantity for the optimal schedule; symbolically,

$$E^{(\text{f},2)}(W_{(\text{ttl})}, n) \;\geq\; E^{(\text{f},2)}(W_{(\text{ttl})}, \text{Algorithm 1}(n)).$$

We now invoke the series of transformations illustrated in Fig. 4 to show that the expected work production of the optimal schedule that partitions each computer's workload into n chunks is no greater than the expected work production of Algorithm 1 that partitions each computer's workload into $2n+1$ chunks. Each of these transformations can not decrease the expected work production.

We begin—see Fig. 4(a)—with an optimal schedule Σ that processes work in n chunks and that satisfies the three properties of Theorem 4. First—see Fig. 4(b)—we transform Σ to schedule Σ_1, by adding a possibly empty $(n+1)$th chunk to the workload of each computer so that each computer processes the entire workload. Clearly, this transformation cannot decrease expected work production. Next—see Fig. 4(c)—we transform Σ_1 to schedule Σ_2 by subdividing chunks so that both computers' chunk boundaries coincide. Formally, let $\mathcal{B}_1$ (resp., $\mathcal{B}_2$) be the set of the "places" in the workload at which there is the boundary of a chunk of computer P_1 (resp., of computer P_2): $\mathcal{B}_1 = \bigcup_{i=0}^{n+1} \left\{ \sum_{j=1}^{i} \omega_{1,j} \right\}$ (resp., $\mathcal{B}_2 = \bigcup_{i=0}^{n+1} \left\{ W_{(\text{ttl})} - \sum_{j=1}^{i} \omega_{2,j} \right\}$). We take the union of these two sets and order the resulting set's elements:

$$\mathcal{B}_1 \cup \mathcal{B}_2 = \{b_1, \ldots, b_l\} \text{ with } 0 = b_0 < b_1 < b_2 < \ldots < b_{l-1} < b_l = W_{(\text{ttl})}.$$

Finally, we specify the new chunks by partitioning the entire workload into l chunks such that:

$$\mathcal{W}'_{1,i} = \mathcal{W}'_{2,l-i+1}, \text{ where } \omega'_{1,i} = b_i - b_{i-1}.$$

We then remark that $l \leq 2n+1$. Indeed, each of computers P_1 and P_2 creates at most n chunk boundaries, which are strictly between 0 and $W_{(\text{ttl})}$. Therefore, in the new schedule, there are at most $2n$ chunk boundaries, each strictly between 0 and $W_{(\text{ttl})}$. This leaves us with at most $2n+1$

chunks, with the boundaries of the whole workload. Finally, we remark that subdividing chunks does not decrease the overall expected work production.

To move closer to the schedule produced by Algorithm 1, we replace the l chunks we just created by l equal-size chunks:

$$(\forall i \in [1,l])\ \ \mathcal{W}''_{1,i} = \mathcal{W}''_{2,l-i+1} \ = \ \left[(i-1)\frac{W_{(\mathrm{ttl})}}{l}, i\frac{W_{(\mathrm{ttl})}}{l}\right].$$

We prove that this indeed gives us a better solution by proving a more general result.

Consider the following schedule-optimization problem for two computers, P_1 and P_2. For $i = 1, 2$, computer P_i executes l_i chunks of possibly different sizes; specifically, it executes chunks $\mathcal{V}_{i,1}, \ldots, \mathcal{V}_{i,l_i}$, in that order. Suppose that P_1 and P_2 have two consecutive chunks in common; i.e., for some $i \in [1, l_1 - 1]$ and $j \in [1, l_2 - 1]$, $\mathcal{V}_{1,i} = \mathcal{V}_{2,j+1}$ and $\mathcal{V}_{1,i+1} = \mathcal{V}_{2,j}$.[10] What should the relative sizes of $\mathcal{V}_{1,i}$ $(=\mathcal{V}_{2,j+1})$ and $\mathcal{V}_{1,i+1}$ $(=\mathcal{V}_{2,j})$ be? To answer this question, we begin with the indicated schedules of P_1 and P_2, and we consider the impact on the overall work expectation of possibly redistributing between the chunks the $|\mathcal{V}_{1,i}| + |\mathcal{V}_{1,i+1}|$ units of work allocated to chunks $\mathcal{V}_{1,i}$ and $\mathcal{V}_{1,i+1}$.

$$\mathcal{E} = |\mathcal{V}_{1,i}|\left(1 - \left(\sum_{k=1}^{i}|\mathcal{V}_{1,k}|\kappa\right)\left(\sum_{k=1}^{j+1}|\mathcal{V}_{2,k}|\kappa\right)\right) + |\mathcal{V}_{1,i+1}|\left(1 - \left(\sum_{k=1}^{i+1}|\mathcal{V}_{1,k}|\kappa\right)\left(\sum_{k=1}^{j}|\mathcal{V}_{2,k}|\kappa\right)\right).$$

To simplify formulas, we use the following abbreviations: $V_1 = \sum_{k=1}^{i-1}|\mathcal{V}_{1,k}|$, $V_2 = \sum_{k=1}^{j-1}|\mathcal{V}_{2,k}|$, and $L = |\mathcal{V}_{1,i}| + |\mathcal{V}_{1,i+1}|$. Then,

$$\mathcal{E} = |\mathcal{V}_{1,i}|\,(1 - (V_1 + |\mathcal{V}_{1,i}|)\kappa(V_2 + L)\kappa) + (L - |\mathcal{V}_{1,i}|)\,(1 - (V_1 + L)\kappa(V_2 + L - |\mathcal{V}_{1,i}|)\kappa)\,.$$

Therefore, the contribution of these two chunks to the expectation is given by:

$$\mathcal{E} = -(V_1 + V_2 + 2L)\kappa^2|\mathcal{V}_{1,i}|^2 + (V_1 + V_2 + 2L)L\kappa^2|\mathcal{V}_{1,i}| + L(1 - (V_1 + L)(V_2 + L)\kappa^2).$$

This expression is maximized by setting $|\mathcal{V}_{1,i}| = \frac{1}{2}L$, that is, by making both chunks have the same size.

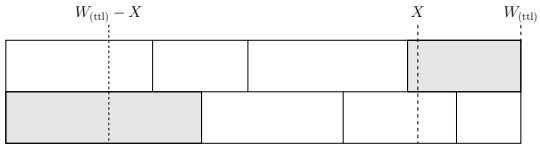
Proceeding by induction, we thus see that replacing l coinciding chunks by l equal-size chunks does not decrease the expectation.

Finally, to obtain a well defined bound using the schedule of Algorithm 1, we enlarge the number of (equal-size) chunks, going from l to $2n+1$. To prove that this last transformation does not decrease the overall expectation, we explicitly calculate the expectation of a solution with n equal-size chunks per computer and show that this expectation is nondecreasing in n.

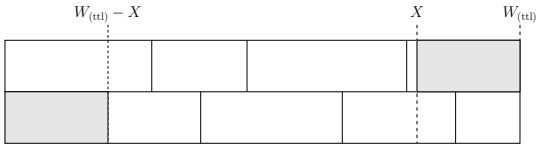
$$\begin{aligned} E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \text{Algorithm 1}(n)) &= \sum_{i=1}^{n}\frac{W_{(\mathrm{ttl})}}{n}\left(1 - \sum_{j=1}^{i}\frac{W_{(\mathrm{ttl})}}{n}\kappa\sum_{j=1}^{n-i+1}\frac{W_{(\mathrm{ttl})}}{n}\kappa\right) \\ &= W_{(\mathrm{ttl})} - \frac{W^3_{(\mathrm{ttl})}}{n^3}\kappa^2\sum_{i=1}^{n}(i(n+1-i)) \\ &= W_{(\mathrm{ttl})} - \frac{W^3_{(\mathrm{ttl})}\kappa^2}{6}\left(1 + \frac{3}{n} + \frac{2}{n^2}\right). \end{aligned}$$

The last expression is obviously nondecreasing in n.

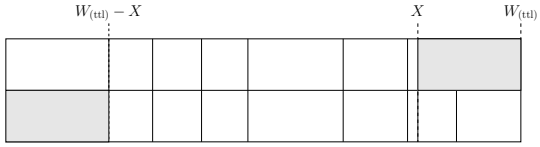
[10] Theorem 4 tells us that P_1 and P_2 should execute these chunks in opposite orders.



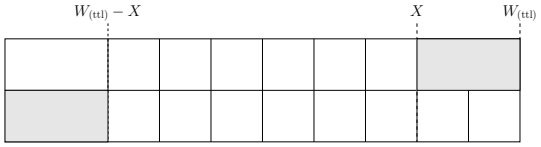
(a) Optimal scheduling with n chunks.



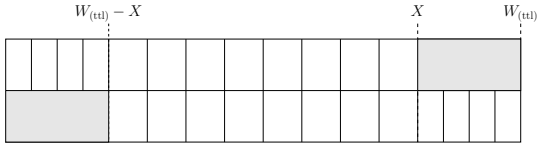
(b) Each workload is extended up to a size X with an $(n+1)$-th chunk.



(c) Dividing chunks for chunk boundaries to coincide.



(d) In each of the three main parts, equalizing the size of chunks.



(e) Solution of Algorithm 1 with $3(n+1)$ chunks.

Figure 5: Series of schedule transformations to prove the asymptotic optimality of Algorithm 1 when $X < W_{(\mathrm{ttl})} < 2X$.

We have thus proved that, for any positive n:

$$E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \text{Algorithm 1}(2n+1)) \;\geq\; E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, n) \;\geq\; E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \text{Algorithm 1}(n)).$$

The optimal expectation is obviously a nondecreasing function bounded above by $W_{(\mathrm{ttl})}$, so that the process converges. Because of the preceding inequality, the optimal expectation has the same limit as does the expectation of Algorithm 1. The latter expectation is thus asymptotically optimal.

3. Case: $X < W_{(\mathrm{ttl})} < 2X$.

As for the previous case, we must prove two things: that the proposed schedule is asymptotically optimal and that its expectation for $\mathsf{W}_{(\mathrm{cmp})}$ is what we claim it is. Let us take any positive integer n. Then, the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ under Algorithm 1 is no greater than this expectation under the optimal scheduling:

$$E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, n) \geq E^{(\mathrm{f},2)}(W_{(\mathrm{ttl})}, \text{Algorithm 1}(n)).$$

Following the series of transformations illustrated by Figure 5, we show that the optimal scheduling with n chunks is not a better expectation than the solution of Algorithm 1 with $3(n+1)$ chunks. Each transformation is a non-decreasing transformation from the point of view of the expectation of $\mathsf{W}_{(\mathrm{cmp})}$.

We start from an optimal scheduling for n chunks satisfying Theorem 4 (see Figure 5(a)). By definition of X any computer-workload no smaller than X is obviously strictly suboptimal. In the first transformation (see Figure 5(b)) we add a $(n+1)$-th chunk to the workload of each computer for each computer-workload to be exactly equal to X. Obviously, this transformation does not change the expectation. Then, we subdivide the chunks so that the boundaries of the chunks of a computer coincide with the boundaries of the chunks of the other computer (see Figure 5(c)). For a formal description of this process, see the proof of the case $W_{(\mathrm{ttl})} \leq X$. Subdividing chunks does not decrease the expectation. We must still count how many chunks we may have in each of

the three main parts of the workload, that is, in the intervals $[0, W_{(\text{ttl})} - X]$, $[W_{(\text{ttl})} - X, X]$, and $[X, W_{(\text{ttl})}]$. Note that each of the interval bounds is a chunk boundary. The chunk boundaries in $[0, W_{(\text{ttl})} - X]$ can only come from original chunks of P_1 and from the bound of the $(n+1)$-th chunk we added to P_2 (which gives a bound at $W_{(\text{ttl})} - X$). Therefore, there are at most $n+1$ chunks in $[0, W_{(\text{ttl})} - X]$. The same is true for $[X, W_{(\text{ttl})}]$. Now, looking at the interval $[W_{(\text{ttl})} - X, X]$, all the boundaries of the chunks $\mathcal{W}_{1,2}$, ..., $\mathcal{W}_{1,n}$ could lie strictly in this interval. The same thing is true for the chunks $\mathcal{W}_{2,2}$, ..., $\mathcal{W}_{2,n}$. Therefore, in the worst case, there can be $2n$ chunk boundaries strictly between $W_{(\text{ttl})} - X$ and X. This gives us at most $2n+1$ chunks in the interval $[W_{(\text{ttl})} - X, X]$. Algorithm 1(3(n+1)) builds a solution with $n+1$ chunks in the interval $[0, W_{(\text{ttl})} - X]$, $2n+2$ chunks in the interval $[W_{(\text{ttl})} - X, X]$, and $n+1$ chunks in the interval $[X, W_{(\text{ttl})}]$. Therefore, in none of the three main parts of the schedule does it have fewer chunks than the solution we just built.

The third transformation is done interval per interval. In each of the intervals $[0, W_{(\text{ttl})} - X]$, $[W_{(\text{ttl})} - X, X]$, and $[X, W_{(\text{ttl})}]$ we distribute the interval workload equally among the chunks (see Figure 5(d)). From the proof of the case $W_{(\text{ttl})} \leq X$ we know that this transformation does not decrease the expectation when it is solely applied to the interval $[W_{(\text{ttl})} - X, X]$. Therefore, what only remains to prove is that this transformation when solely applied to the interval $[0, W_{(\text{ttl})} - X]$ does not decrease the expectation (the fact that the different intervals do not impact each other is due to our failure model and to the fact that no chunk simultaneously strictly belongs to two intervals). To establish the desired result we only consider two consecutive chunks of P_1, $\mathcal{V}_{1,i}$ and $\mathcal{V}_{1,i+1}$ belonging in the interval $[0, W_{(\text{ttl})} - X]$ (and thus which do not intersect the workload of P_2). The contribution of these two chunks to the expectation is:

$$\mathcal{E} = |\mathcal{V}_{1,i}| \left(1 - \sum_{j=1}^{i} |\mathcal{V}_{1,j}|\kappa\right) + |\mathcal{V}_{1,i+1}| \left(1 - \sum_{j=1}^{i+1} |\mathcal{V}_{1,j}|\kappa\right).$$

Using the notations $L = |\mathcal{V}_{1,i}| + |\mathcal{V}_{1,i+1}|$ and $V = \sum_{j=1}^{i-1} \mathcal{V}_{1,j}$, we have:

$$\begin{aligned} \mathcal{E} &= |\mathcal{V}_{1,i}| \left(1 - (V + |\mathcal{V}_{1,i}|)\kappa\right) + (L - |\mathcal{V}_{1,i}|) \left(1 - (V + L)\kappa\right) \\ &= -|\mathcal{V}_{1,i}|^2\kappa + L|\mathcal{V}_{1,i}|\kappa + L(1 - (V + L)\kappa). \end{aligned}$$

This last expression is obviously maximized when $|\mathcal{V}_{1,i}| = \frac{L}{2}$, that is, when the two consecutive chunks have the same size.

The last transformation increases the number of same-size jobs in each of the three phases of the scheduling for these numbers to respectively be $n+1$, $2(n+1)$, and $n+1$ (see Figure 5(b)). We already know, from the study of the case $W_{(\text{ttl})} \leq X$, that this is not decreasing the expectation for the chunks in the interval $[W_{(\text{ttl})} - X, X]$. We now show that this is also the case for chunks in the interval $[0, W_{(\text{ttl})} - X]$. The cumulative expectation for the m equal-size chunks of the interval $[0, W_{(\text{ttl})} - X]$ is:

$$\begin{aligned} \mathcal{E} &= \sum_{i=1}^{m} \frac{W_{(\text{ttl})} - X}{m} \left(1 - \sum_{j=1}^{i} \frac{W_{(\text{ttl})} - X}{m}\kappa\right) \\ &= (W_{(\text{ttl})} - X) - \frac{(W_{(\text{ttl})} - X)^2\kappa}{m^2} \sum_{i=1}^{m} i \\ &= (W_{(\text{ttl})} - X) - \left(\frac{1}{2} + \frac{1}{2m}\right) (W_{(\text{ttl})} - X)^2\kappa \end{aligned}$$

which is obviously increasing with m.

The expectation of $\mathsf{W}_{(\text{cmp})}$, with $l = \lfloor \frac{n}{3} \rfloor$, for the scheduling of Algorithm 1(n) is then equal

to:

$$\begin{aligned}
\mathcal{E} = & \sum_{i=1}^{l} \frac{W_{(\mathrm{ttl})} - X}{l} \left(1 - i\frac{W_{(\mathrm{ttl})} - X}{l}\kappa\right) \\
+ & \sum_{i=1}^{2l} \frac{2X - W_{(\mathrm{ttl})}}{2l} \left(1 - \left(W_{(\mathrm{ttl})} - X + i\frac{2X - W_{(\mathrm{ttl})}}{2l}\right)\kappa \right. \\
& \qquad \left. \times \left(X - (i-1)\frac{2X - W_{(\mathrm{ttl})}}{2l}\right)\kappa\right) \\
+ & \sum_{i=1}^{l} \frac{W_{(\mathrm{ttl})} - X}{l} \left(1 - i\frac{W_{(\mathrm{ttl})} - X}{l}\kappa\right) \\
= & \; 2W_{(\mathrm{ttl})} - \frac{1}{3}X - W_{(\mathrm{ttl})}^2\kappa + \frac{W_{(\mathrm{ttl})}^3\kappa^2}{6} \\
+ & \frac{1}{l}\left(\left(1 + \frac{1}{l}\right)W_{(\mathrm{ttl})} - \left(1 + \frac{2}{3l}\right)X - \frac{1}{2l}W_{(\mathrm{ttl})}^2\kappa \right. \\
- & \left. \frac{1}{4}\left(1 - \frac{1}{3l}\right)W_{(\mathrm{ttl})}^3\kappa^2\right).
\end{aligned}$$

□

5 Scheduling for p Remote Computers

We finally turn to the general case, wherein there are p remote computers. We have discovered this case of general p to be much more difficult than the already challenging case $p = 2$, so we devote our efforts here to searching for *efficient heuristic schedules*.

> In order to appreciate how hard it is to extend the case $p = 2$ even to $p = 3$, the reader is invited to seek an analogue of Theorem 4 for $p = 3$. As one example, we have not discovered a 3-computer analogue of "mirroring," and our attempts to do so have all fallen to unobvious schedules such as those discussed after Theorem 5.

Because of the difficulty of the general scheduling problem, we adopt a pragmatic approach, by focusing only on the linear risk model, and by restricting attention to "well-structured" schedules that employ same size chunks.

> Our restriction to same-size chunks has two major antecedents. (1) The optimal schedules for the case $p = 1$ and the asymptotically optimal schedules for the case $p = 2$ mandate using same-size chunks. This suggests that such chunking may be computationally beneficial. (2) This restriction greatly simplifies the specification and implementation of schedules for the case of general p, by imposing simplifying structure on this extremely hard scheduling problem.

All of the schedules we develop here operate as follows.

1. They *partition* the total workload into (disjoint) *slices* that they assign to—and replicate on—disjoint subsets (*coteries*) of remote computers. (Each computer partitions each slice into *same-size* chunks.)

2. They orchestrate the processing of the slices on each coterie of remote computers.

5.1 The Partitioning Phase

We begin with some simple partitioning heuristics that are tailored to the linear risk function—but we suggest how they can be adapted to other risk functions. We partition our scheduling problem into three subproblems, based on the size of the workload we wish to schedule. This partition—which acknowledges the futility of deploying a workslice of size $> X$ on any computer,

in the light of our interruption model—gives us one easy subproblem and two challenging ones that will occupy the rest of our attention.

$W_{(\text{ttl})}$ is "very large." When $W_{(\text{ttl})} \geq pX$, we deploy p slices of common size X, to be processed independently on the remote computers. We abandon the remaining $W - pX$ units of work, in acknowledgment of our interruption model. (We assume here that work is not prioritized, so we do not care *which* pX units we deploy.) We then reuse on each computer the results of Section 3.

$W_{(\text{ttl})}$ is "very small." When $W_{(\text{ttl})} \leq X$, we deploy the entire workload in a single slice, which we replicate on all p computers. The partitioning phase is therefore obvious; all the scheduling work is done in the orchestration phase.

$W_{(\text{ttl})}$ is of "intermediate" size. The case $X < W_{(\text{ttl})} < pX$ is the interesting challenge, as there is no compelling scheduling strategy. There is no point of deploying more than X units of work on a single computer. Therefore, we decide to deploy $W_{(\text{dpl})} = \min(W_{(\text{ttl})}, pX)$ units of work in to the p remote computers. The p computers enable to replicate each "piece of work" $pX/W_{(\text{dpl})}$ times on average. For the sake of simplicity, we have disjoint coteries of computers working on independent slices of work. Ideally, we would like to have $p/\frac{pX}{W_{(\text{dpl})}} = W_{(\text{dpl})}\kappa$ coteries. As we must have an integer number of computers in each coterie, we partition the work into $q = \lceil Z\kappa \rceil$ slices. We balance computing resources as much as possible, by replicating each slice on either $\lfloor p/q \rfloor$ or $\lceil p/q \rceil$ remote computers. To balance the load among the coteries, a coterie with $\lfloor p/q \rfloor$ computers will work on a slice of size $\mathsf{sl}_- = \lfloor p/q \rfloor \frac{Z}{p}$, and a coterie with $\lceil p/q \rceil$ computers will work on a slice of size $\mathsf{sl}_+ = \lceil p/q \rceil \frac{Z}{p}$.

> Among the ways in which we have tailored the preceding scenario to the linear risk function is by demanding that the load of a single computer has a size $\leq X$. For general risk functions, we would introduce a parameter λ that specifies the maximum probability of interruption that the user would allow for the work allocated to a computer. For linear risk functions we used $\lambda = 1$, but this choice would be impractical, for instance, for heavy tailed distributions. Hence the need for the λ parameter. We would then use λ to compute the maximum work maxsl allocatable to a computer by insisting that $Pr(\mathsf{maxsl}) = \lambda$. For instance if $\lambda = 1/2$, then with the linear risk function we would set $\mathsf{maxsl} = \frac{1}{2}X$, while with the exponential risk function we would set $\mathsf{maxsl} = (\ln 2)X$. The amount of work we actually deploy would now be $W_{(\text{dpl})} = \min(W_{(\text{ttl})},\ p \times \mathsf{maxsl})$. This would mandate using $q = \lceil Z/\mathsf{maxsl} \rceil$ slices, of sizes defined as previously.

We now specify the partition procedure, Algorithm 2, which takes three inputs: the total amount of work $W_{(\text{ttl})}$, the number p of computers, and the maximum allowable risk λ. The algorithm returns the number of slices, their sizes, and the number of remote computers that each slice is deployed to.

Algorithm 2: The partitioning algorithm for p computers.

```
1  procedure Partition(W_(ttl), p, λ)
2  begin
3      /*Determine maxsl such that Pr(maxsl) = λ*/
4      W_(dpl) ← min(W_(ttl), p × maxsl)
5      q ← ⌈Z/maxsl⌉
6      sl_− ← ⌊p/q⌋ Z/p, sl_+ ← ⌈p/q⌉ Z/p
7      r ← p mod q; s ← q − r
8      Partition the computers into:
9          r coteries of cardinality ⌊p/q⌋ + 1 each and working on slices of size sl_+, and
10         s coteries of cardinality ⌊p/q⌋ each and working on slices of size sl_−.
11 end
```

5.2 The Orchestration Phase

The partition phase has left us with independent slices of work that will be executed by disjoint coteries of computers. A slice, of size sl, will be partitioned into n chunks of common size $\omega = \mathsf{sl}/\mathsf{n}$, where the "checkpointing granularity" n is supposed to be given to us (cf. Section 5.3). For each coterie Γ of computers, each chunk assigned to coterie Γ will be executed by all $\mathsf{g}_\Gamma \in \{\lfloor p/q \rfloor\,,\ 1 + \lfloor p/q \rfloor\}$ computers in the coterie. Our challenge is to determine how to orchestrate the g_Γ executions of each chunk—i.e., to determine when (at which time step) and where (on which computer) to execute which chunk—in a way that maximizes the expected amount of work completed by the total assemblage of p computers. The remainder of our study is dedicated to this orchestration phase.

5.2.1 General schedules

Let us motivate our approach to the orchestration problem via the following example, wherein each slice is partitioned into $\mathsf{n} = 12$ chunks, and each coterie contains $\mathsf{g} = 4$ computers. Since each coterie of computers operates independently of all others, we can specify the overall schedule coterie by coterie. For each coterie Γ and its associated slice, we represent a possible schedule for Γ's executing the slice via a table such as Table 1; we call these tables *execution charts*. Rows in these charts enumerate the computers in the associated coterie Γ, and columns enumerate the indices of the chunks into which coterie Γ's slice is chopped. Chart-entry $C_{i,j}$ is the step at which chunk j is processed by computer P_i.

Computer \ Chunk	1	2	3	4	5	6	7	8	9	10	11	12
P_1	1	6	9	12	2	5	8	11	3	4	7	10
P_2	12	1	6	9	11	2	5	8	10	3	4	7
P_3	9	12	1	6	8	11	2	5	7	10	3	4
P_4	6	9	12	1	5	8	11	2	4	7	10	3

Table 1: An execution chart for a coterie of four computers. In this example, chunk 5 is executed by P_2 at step $C_{2,5} = 11$.

Any $\mathsf{g} \times \mathsf{n}$ integer matrix whose rows are permutations of $[1..\mathsf{n}]$ can be used as the execution chart for a valid schedule for the slice, under which each P_i executes once each chunk j (specifically, at step $C_{i,j}$). One can use such a chart to calculate the expected amount of work completed under the schedule that the chart specifies. To wit, chunk j will *not* be executed under a schedule only if all g computers in the coterie are interrupted before they complete the chunk. This occurs with probability

$$\prod_{i=1}^{\mathsf{g}} Pr(C_{i,j}\omega) \;=\; \prod_{i=1}^{\mathsf{g}} Pr\left(C_{i,j}\mathsf{sl}/\mathsf{n}\right),$$

so the expectation of the total work completed from the slice is

$$\begin{aligned} E(\mathsf{sl},\mathsf{n}) &= (\mathsf{sl}/\mathsf{n}) \sum_{j=1}^{\mathsf{n}} \left(1 - \prod_{i=1}^{\mathsf{g}} Pr\left(C_{i,j}\mathsf{sl}/\mathsf{n}\right)\right) \\ &= \mathsf{sl}\left(1 - \frac{1}{\mathsf{n}}\left(\frac{\mathsf{sl}\kappa}{\mathsf{n}}\right)^{\mathsf{g}} \sum_{j=1}^{\mathsf{n}} \prod_{i=1}^{\mathsf{g}} C_{i,j}\right). \end{aligned} \tag{16}$$

The last expression, (16), is specific to the linear-risk model and assume that $C_{i,j}\omega\kappa \le 1$ or, equivalently, that $Pr(C_{i,j}\omega) = C_{i,j}\omega\kappa$. We will make this assumption in the remaining of this

section, each time we will write an expectation. [11] We can, therefore, derive the following upper bound:

Proposition 1.

$$E(\mathsf{sl},\mathsf{n}) \;\leq\; E_{\max} \;=\; \mathsf{sl}\cdot\left(1-\left(\mathsf{sl}\cdot\kappa\frac{(\mathsf{n}!)^{1/\mathsf{n}}}{\mathsf{n}}\right)^{\mathsf{g}}\right).$$

Proof. Let $\mathsf{cp}_j = \prod_{i=1}^{\mathsf{g}} C_{i,j}$ be the j-th column product in the chart. From the expression of $E(\mathsf{sl},\mathsf{n})$, we see that it is maximum when the sum of the n column products is minimum. But the product of the column products is constant, because each row is a permutation of $[1..\mathsf{n}]$: we have $\prod_{j=1}^{\mathsf{n}} \mathsf{cp}_j = (\mathsf{n}!)^{\mathsf{g}}$. The sum is minimum when all products are equal (to $(\mathsf{n}!)^{\frac{\mathsf{g}}{\mathsf{n}}}$), whence the result. □

Stirling's formula gives a useful approximation of the upper bound when n is large:

$$E_{\max} \approx \mathsf{sl}\cdot\left(1-\left(\frac{\mathsf{sl}.\kappa}{e}\right)^{\mathsf{g}}\right).$$

5.2.2 Group schedules: introduction

Referring back to Table 1, we observe that chunks 1, 2, 3, and 4 are always executed at the same steps, by different computers; the same is true for chunks $5, 6, 7, 8$ as a group, and for chunks $9, 10, 11, 12$ as a group. The twelve chunks of the slice thus partition naturally into three *groups*. By respecifying the schedule of Table 1 as the *group(-oriented)* schedule of Table 2, we significantly

Group 1 chunks 1–4	Group 2 chunks 5–8	Group 3 chunks 9–12
1	2	3
6	5	4
9	8	7
12	11	10

Table 2: Execution chart for a group-oriented schedule. Rows represent time steps for the first computer in each group associated with each column; the remaining computers' schedules are obtained by cyclic downward permutations of the rows.

simplify the specification. Note that the meanings of rows and columns have changed in this reorientation: compare Tables 1 and 2 as we describe the changes. In the group(-oriented) execution chart of Table 2, each column corresponds to a group of chunks; entry (i, j) of the chart specifies the step at which each computer executes its ith chunk within group j. The schedule for computer P_j, where $j \in \{2, 3, 4\}$, is obtained by cyclically permuting (downward) the schedule for P_1 $j-1$ times. The important feature here is that this orchestration has each computer attempt to execute each chunk exactly once.

We generalize this description. When n is a multiple of g, we can sometimes convert the full $\mathsf{g}\times\mathsf{n}$ execution chart C, as exemplified by Table 1, to the $\mathsf{g}\times\mathsf{n}/\mathsf{g}$ *group(-oriented)* execution chart $\widehat{C}$ exemplified by Table 2. There are n/g groups, each of size g, and chart-entry $\widehat{C}_{i,j}$ denotes the step at which group j of chunks is executed for the ith time. It is tacitly assumed that chunk-indices within each group are cyclically permuted (downward) at each step, so that each chunk ends up being processed by each computer. Thus, in order for a chart $\widehat{C}$ to specify a valid group

[11] Under our partitioning scheme, this assumption is true for any coterie except the 2-computer coteries when $X < W_{(\mathrm{ttl})} < 2X$. Taking this case into account, however, would considerably complicate all the expectation formulas and would forbid us to make any conclusion when comparing heuristics. Furthermore, the conclusions we reach using this assumption—mainly those derived from Table 6—are backed by the experiments of Section 6 which consider all cases. These experiments provide an a posteriori justification for our simplifying assumption.

schedule, its total set of entries must be a permutation of $[1..n]$. When $\widehat{C}$ does specify a valid group schedule, the expected amount of work it completes, under the linear risk model, is:

$$E(\mathsf{sl},\mathsf{n}) \;=\; \mathsf{sl}\left(1-\frac{\mathsf{g}}{\mathsf{n}}\left(\frac{\mathsf{sl}\cdot\kappa}{\mathsf{n}}\right)^{\mathsf{g}}\sum_{j=1}^{\mathsf{n}/\mathsf{g}}\prod_{i=1}^{\mathsf{g}}\widehat{C}_{i,j}\right). \tag{17}$$

The preceding expression exposes the importance of the constant

$$\mathsf{K}^{(\Sigma)} \;=\; \sum_{j=1}^{\mathsf{n}/\mathsf{g}}\prod_{i=1}^{\mathsf{g}}\widehat{C}^{(\Sigma)}_{i,j}$$

as a measure of a group schedule Σ's performance; to wit,

$$E(\mathsf{sl},\mathsf{n},\Sigma) \;=\; \mathsf{sl}-\mathsf{K}^{(\Sigma)}\cdot\frac{\mathsf{g}}{\kappa}\left(\frac{\mathsf{sl}\kappa}{\mathsf{n}}\right)^{\mathsf{g}+1}. \tag{18}$$

Thus: *A smaller value of* $\mathsf{K}^{(\Sigma)}$ *corresponds to a larger value of* $E(\mathsf{sl},\mathsf{n},\Sigma)$.

Group schedules are very natural, because they are *symmetric:* all computers play the same role as the work is processed, differing only in the times at which they process different chunks. Intuition suggests that the most productive schedules are symmetric: why should some of the identical computers be treated differently by "nature" than others? Indeed, the following upper bound on the expected work production of group schedules—which is the best we have been able to prove—does not distinguish symmetric schedules from general ones—but we have not yet been able to prove that no difference exists.

Proposition 2. *For any group schedule* Σ*,*

$$E(\mathsf{sl},\mathsf{n},\Sigma) \;\le\; E_{\max} \;=\; \mathsf{sl}-\frac{\mathsf{sl}^{\mathsf{g}+1}\kappa^{\mathsf{g}}(\mathsf{n}!)^{\mathsf{g}/\mathsf{n}}}{\mathsf{n}^{\mathsf{g}}}.$$

Proof. Let $\mathsf{cp}_j=\prod_{i=1}^{\mathsf{g}}G_{i,j}$ be the j-th column product. As before, $E(\mathsf{sl},\mathsf{n})$ is maximum when the sum of the $\frac{\mathsf{n}}{\mathsf{g}}$ column products is minimum. The product of these column products is equal to a constant $(\mathsf{n}!)$. The sum is minimum when all products are equal to $(\mathsf{n}!)^{\frac{\mathsf{g}}{\mathsf{n}}}$, hence the same result as for Proposition 1. □

Note that Proposition 2 affords us an easy lower bound, $\mathsf{K}_{\min}$ on the K value of any group schedule with the parameters g and n:

$$\mathsf{K}_{\min} \;=\; \left\lceil\frac{\mathsf{n}}{\mathsf{g}}(\mathsf{n}!)^{\mathsf{g}/\mathsf{n}}\right\rceil.$$

5.2.3 Group schedules: specific schedules

Our group schedules strive to maximize expected work completion by having every computer attempt to compute every chunk. Of course, there are many ways to achieve this coverage, and the form of the risk function will make some ways more advantageous than others with respect to maximizing expected work completion. As an extreme example, in the case $p = 2$, for *every* risk function, it is advantageous to have the remote computers process the work they share "in opposite orders" (Theorem 4). We now specify and compare the performance of six group schedules whose chunk-scheduling regimens seem to be a good match for the way the linear risk function "predicts" interruptions. We specify each schedule Σ via its group execution chart $\widehat{C}^{(\Sigma)}$—see Fig. 6—and we represent the performance of each schedule Σ via its performance constant $\mathsf{K}^{(\Sigma)}$. The beneficent structures of these schedules is evidenced by our ability to present explicit symbolic expressions for their K constants.

Group 1	Group 2	Group 3
1	2	3
4	5	6
7	8	9
10	11	12

(a) Cyclic: $\mathsf{K} = 3104$

Group 1	Group 2	Group 3
1	2	3
6	5	4
9	8	7
12	11	10

(b) Reverse: $\mathsf{K} = 2368$

Group 1	Group 2	Group 3
1	2	3
4	5	6
9	8	7
12	11	10

(c) Mirror: $\mathsf{K} = 2572$

Group 1	Group 2	Group 3
1	2	3
6	5	4
7	8	9
12	11	10

(d) Snake: $\mathsf{K} = 2464$

Group 1	Group 2	Group 3
1	2	3
8	6	4
9	7	5
10	11	12

(e) Fat snake: $\mathsf{K} = 2364$

Figure 6: Five group schedules with their associated K values. For this instance, $\mathsf{K}_{\min} = 2348$.

Cyclic scheduling (Fig. 6(a)). Under this simplest scheduling regimen, Σ_{cyclic}, groups are executed sequentially, in a round-robin fashion. Specifically, the chunks of group j are executed at steps j, $j + \mathsf{n}/\mathsf{g}$, $j + 2\mathsf{n}/\mathsf{g}$, and so on. We find that

$$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} = \sum_{j=1}^{\mathsf{n}/\mathsf{g}} \prod_{k=0}^{\mathsf{g}-1} (j + k\mathsf{n}/\mathsf{g}) .$$

The weakness of Σ_{cyclic} is that chunks in low-index groups have a higher probability of being completed successfully than do chunks in high-index groups—because chunks remain in the same relative order throughout the computation. The remaining schedules that we consider aim to compensate for this imbalance via different intuitively motivated strategies.

Reverse scheduling (Fig. 6(b)). A schedule $\Sigma_{\mathsf{reverse}}$ produced under this regimen executes the chunks in each group once in the initially-specified order, and then executes them in the *reverse* order $\mathsf{n}/\mathsf{g} - 1$ times. The schedule thereby strives to compensate for the imbalance in chunks' likelihoods of being completed created by their initial order of processing. ($\Sigma_{\mathsf{reverse}}$ is the schedule specified in Table 2.) Under $\Sigma_{\mathsf{reverse}}$, the chunks in group j are executed at step j, and thereafter at steps $2\mathsf{n}/\mathsf{g} - j + 1$, $3\mathsf{n}/\mathsf{g} - j + 1$, $4\mathsf{n}/\mathsf{g} - j + 1$, and so on. We find that

$$\mathsf{K}^{(\Sigma_{\mathsf{reverse}})} = \sum_{j=1}^{\mathsf{n}/\mathsf{g}} j \times \prod_{k=1}^{\mathsf{g}-1} ((k+1)\mathsf{n}/\mathsf{g} - j + 1) .$$

Mirror scheduling (Fig. 6(c)). The mirror schedule Σ_{mirror}, which is defined only when g is even, represents a compromise between the cyclic and reverse scheduling strategies. Σ_{mirror} compensates for the imbalance in likelihood of completion only during the second half of the computation. Specifically, Σ_{mirror} mimics Σ_{cyclic} for the first $\mathsf{g}/2$ phases of processing a group, and it mimics

$\Sigma_{\mathsf{reverse}}$ for the second $\mathsf{g}/2$ phases. We find that

$$\mathsf{K}^{(\Sigma_{\mathsf{mirror}})} = \sum_{j=1}^{\mathsf{n}/\mathsf{g}} \prod_{k=0}^{\frac{1}{2}\mathsf{g}-1} (j + k\mathsf{n}/\mathsf{g})\,((p-k)\mathsf{n}/\mathsf{g} - j + 1)\,.$$

Snake-like scheduling (Fig. 6(d)). Our fourth schedule, Σ_{snake}, compensates for the imbalance of the cyclic schedule by mimicking Σ_{cyclic} at every odd-numbered step and mimicking $\Sigma_{\mathsf{reverse}}$ at every even-numbered step, thereby lending a snake-like structure to the execution chart $\widehat{C}^{(\Sigma_{\mathsf{snake}})}$. We find that

$$\mathsf{K}^{(\Sigma_{\mathsf{snake}})} = \sum_{j=1}^{\mathsf{n}/\mathsf{g}} \prod_{k=0}^{\frac{1}{2}\mathsf{g}-1} (j + 2k\mathsf{n}/\mathsf{g})\,(2(k+1)\mathsf{n}/\mathsf{g} - j + 1)\,.$$

Fat snake-like scheduling (Fig. 6(e)). Our final, fifth schedule, $\Sigma_{\mathsf{fat-snake}}$, qualitatively adopts the same strategy as does Σ_{snake}, but it slows down the return phase of the latter schedule. Consider, for illustration, three consecutive rows of $\widehat{C}^{(\Sigma_{\mathsf{fat-snake}})}$. The first row is identical to its shape in $\widehat{C}^{(\Sigma_{\mathsf{cyclic}})}$. The return phase of Fat snake distributes elements of the two remaining rows in the reverse order, two elements at a time. The motivating intuition is that the slower return would further compensate for the imbalance in Σ_{cyclic}. We find that

$$\mathsf{K}^{(\Sigma_{\mathsf{fat-snake}})} = \sum_{j=0}^{\mathsf{n}/\mathsf{g}-1} \prod_{k=0}^{\frac{1}{3}\mathsf{g}-1} (1 + j + 3k\mathsf{n}/\mathsf{g})\,(3(k+1)\mathsf{n}/\mathsf{g} - 2j - 1)\,(3(k+1)\mathsf{n}/\mathsf{g} - 2j)\,.$$

We derive the following performance bounds for these five schedules

Proposition 3. *The values of* $\mathsf{K}^{(\Sigma)}$ *for our five scheduling algorithms satisfy the following lower and upper bounds:*

$$\begin{aligned}
\frac{1}{\mathsf{n}} &\le \frac{\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})}}{\mathsf{g}!\,(\mathsf{n}/\mathsf{g})^{\mathsf{g}+1}} \le 1 \\
\frac{1}{2\mathsf{g}} &\le \frac{\mathsf{K}^{(\Sigma_{\mathsf{reverse}})}}{\mathsf{g}!\,(\mathsf{n}/\mathsf{g})^{\mathsf{g}+1}} \le \tfrac{1}{2}(\mathsf{n}+\mathsf{g}) \\
\frac{1}{\mathsf{n}} &\le \frac{\mathsf{K}^{(\Sigma_{\mathsf{mirror}})}}{\mathsf{g}!\,(\mathsf{n}/\mathsf{g})^{\mathsf{g}+1}} \le 1 \\
\frac{\mathsf{g}}{\mathsf{n}^2} &\le \frac{\mathsf{K}^{(\Sigma_{\mathsf{snake}})}}{\mathsf{g}!\,(\mathsf{n}/\mathsf{g})^{\mathsf{g}+1}} \le 1 \\
\frac{1}{(\mathsf{g}-1)\mathsf{n}} &\le \frac{\mathsf{K}^{(\Sigma_{\mathsf{fat-snake}})}}{\mathsf{g}!\,(\mathsf{n}/\mathsf{g})^{\mathsf{g}+1}} \le \mathsf{g}
\end{aligned}$$

Proof. The calculations are straightforward. For the Cyclic schedule, we have

$$\mathsf{K}_{\mathsf{cyclic}} = \sum_{j=1}^{\frac{\mathsf{n}}{\mathsf{g}}} \prod_{k=0}^{\mathsf{g}-1} \left(j + k\frac{\mathsf{n}}{\mathsf{g}}\right).$$

We derive the lower bound by replacing index j by 0 in the summation (except in the term $k=0$ where we replace j by 1):

$$\mathsf{K}_{\mathsf{cyclic}} \ge \frac{\mathsf{n}}{\mathsf{g}} \left(\prod_{k=1}^{\mathsf{g}-1} k\frac{\mathsf{n}}{\mathsf{g}}\right) = (\mathsf{g}-1)! \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}} = \frac{1}{\mathsf{n}}\mathsf{g}! \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}+1}.$$

Similarly, we let $j = \frac{\mathsf{n}}{\mathsf{g}}$ in each term of the summation to get the upper bound. We proceed in a similar way for the other three variants.

We explicit the computations for Fat snake, as they are a bit less obvious. For the lower bound we have:

$$\begin{aligned}
\mathsf{K}_{\mathsf{fat-snake}} &= \sum_{j=0}^{\frac{\mathsf{n}}{\mathsf{g}}-1} \prod_{k=0}^{\frac{\mathsf{g}}{3}-1} \left(1 + j + 3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}} - 2j - 1\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}} - 2j\right) \\
&\geq \sum_{j=0}^{\frac{\mathsf{n}}{\mathsf{g}}-1} \prod_{k=0}^{\frac{\mathsf{g}}{3}-1} \left(1 + 3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}} - 2\frac{\mathsf{n}}{\mathsf{g}} + 1\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}} - 2\frac{\mathsf{n}}{\mathsf{g}} + 2\right) \\
&\geq \frac{\mathsf{n}}{\mathsf{g}} \prod_{k=0}^{\frac{\mathsf{g}}{3}-1} \left(1 + 3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+1)\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+1)\frac{\mathsf{n}}{\mathsf{g}}\right) \\
&\geq \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^3 \prod_{k=1}^{\frac{\mathsf{g}}{3}-1} \left(3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+1)\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+1)\frac{\mathsf{n}}{\mathsf{g}}\right) \\
&\geq \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^3 \prod_{k=1}^{\frac{\mathsf{g}}{3}-1} \left((3k-1)\frac{\mathsf{n}}{\mathsf{g}}\right)\left(3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+1)\frac{\mathsf{n}}{\mathsf{g}}\right) \\
&= \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}} (\mathsf{g}-2)!
\end{aligned}$$

For the upper bound we derive:

$$\begin{aligned}
\mathsf{K}_{\mathsf{fat-snake}} &= \sum_{j=0}^{\frac{\mathsf{n}}{\mathsf{g}}-1} \prod_{k=0}^{\frac{\mathsf{g}}{3}-1} \left(1 + j + 3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}} - 2j - 1\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}} - 2j\right) \\
&\leq \sum_{j=0}^{\frac{\mathsf{n}}{\mathsf{g}}-1} \prod_{k=0}^{\frac{\mathsf{g}}{3}-1} \left(\frac{\mathsf{n}}{\mathsf{g}} + 3k\frac{\mathsf{n}}{\mathsf{g}}\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}}\right)\left(3(k+1)\frac{\mathsf{n}}{\mathsf{g}}\right) \\
&\leq \frac{\mathsf{n}}{\mathsf{g}} \prod_{k=0}^{\frac{\mathsf{g}}{3}-1} \left((3k+1)\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+3)\frac{\mathsf{n}}{\mathsf{g}}\right)\left((3k+3)\frac{\mathsf{n}}{\mathsf{g}}\right) \\
&\leq \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}+1} \left(\prod_{k=0}^{\frac{\mathsf{g}}{3}-2} (3k+2)(3k+3)(3k+4)\right)(\mathsf{g}-2)\mathsf{g}^2 \\
&= \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}+1} (\mathsf{g}-2)!(\mathsf{g}-2)\mathsf{g}^2 \leq \left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}+1} (\mathsf{g})!\mathsf{g}
\end{aligned}$$

□

From the size of its bounds on $\mathsf{K}^{(\Sigma_{\mathsf{snake}})}$, Proposition 3 suggests that schedule Σ_{snake} may be the most efficient of the five group-scheduling algorithms we have considered, especially when we checkpoint often, i.e., when n is large. We will evaluate this possibility via the experiments reported at the end of this subsection. While still focusing on mathematical analyses of our schedules, though, we use Stirling's formula to derive more evocative bounds on $\mathsf{K}^{(\Sigma_{\mathsf{snake}})}$: $\mathsf{K}_{\min} \leq \mathsf{K}^{(\Sigma_{\mathsf{snake}})} \leq \mathsf{K}_{\mathsf{upper}}$, where

$$\mathsf{K}_{\min} \approx \frac{e}{g}\left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}+1} \quad \text{and} \quad \mathsf{K}_{\mathsf{upper}} = \mathsf{g}!\left(\frac{\mathsf{n}}{\mathsf{g}}\right)^{\mathsf{g}+1} \approx \frac{e\sqrt{2\pi}}{\sqrt{g}}\left(\frac{\mathsf{n}}{e}\right)^{\mathsf{g}+1}.$$

We conclude this subsection by adding a last element to our set of group schedules. The resulting "greedy" procedure strives to iteratively balance the probability of success for each group of chunks. As we do not get any asymptotic estimation for Greedy, we content ourselves with a numerical estimate.

Greedy scheduling (Table 3). The greedy scheduling algorithm, Σ_{greedy}, iteratively assigns a step to each group of chunks so as to balance the current success probabilities as much as possible.

At each step, Σ_{greedy} constructs one new row of the execution chart $\widehat{C}^{(\mathsf{greedy})}$. Remember that, after k steps, the probability that a chunk in group j will be interrupted is proportional to the product $\Pi_{i=1}^{k}\widehat{C}_{ij}^{(\mathsf{greedy})}$ of the entries in column j of the chart. The idea is to sort current column products and to assign the smallest time-step to the largest product, and so on. Table 3 illustrates a computation with $\mathsf{n} = 12$ and $\mathsf{g} = 4$. In this example, Σ_{greedy} is identical to $\Sigma_{\mathsf{reverse}}$, hence achieves the same performance constant $\mathsf{K}^{(\Sigma_{\mathsf{greedy}})} = \mathsf{K}^{(\Sigma_{\mathsf{reverse}})} = 2368$.

Step 1	1	2	3
CCP	1	2	3
Step 2	6	5	4
CCP	6	10	12
Step 3	9	8	7
CCP	54	80	84
Step 4	12	11	10
CCP	6	880	12

Table 3: A computation by Σ_{greedy}. CCP denotes the *Current Column Product*.

For the record, and for the curious reader: Table 4 provides an example for which none of our group schedules is optimal, and Table 5 shows an example for which Σ_{greedy} differs from, and outperforms, $\Sigma_{\mathsf{reverse}}$.

1 2 3 4 5 6 7 8 9	1 2 3 6 5 4 7 8 9	1 2 3 6 5 4 9 8 7	1 2 3 8 6 4 9 7 5	1 2 3 8 5 4 9 7 6
$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} = 270$	$\mathsf{K}^{(\Sigma_{\mathsf{snake}})} = 230$	$\mathsf{K}^{(\Sigma_{\mathsf{reverse}})} = \mathsf{K}^{(\Sigma_{\mathsf{greedy}})} = 218$	$\mathsf{K}^{(\Sigma_{\mathsf{fat\ snake}})} = 216$	$\mathsf{K}_{\mathsf{optimal}} = \mathsf{K}_{\min} = 214$

Table 4: Comparing group schedules for $\mathsf{n} = 9$ and $\mathsf{g} = 3$. (Σ_{mirror} is missing because g is odd). Here $\Sigma_{\mathsf{reverse}}$ and Σ_{greedy} are identical. The optimal schedule achieves the bound $\mathsf{K}_{\min}$.

Numerical evaluation. We ran all six of our scheduling heuristics on all problems where $g \in [2, 100]$, $n \in [2{*}g, 1000]$, and g divides n; altogether, this corresponds to 4032 instances. We report in Table 6 two series of statistics. In the *Relative* series, we form the ratio of the K value of a given heuristic on a given instance over the lowest K value found for that instance among all the tested heuristics. For the *Absolute* series, we form the ratio with $\mathsf{K}_{\min}$. In Table 6 we also report the *best-of* heuristic that, on each instance, runs the six other algorithms and picks the best answer.

Σ_{greedy} is clearly the best heuristic: it finds the best schedule for 83% of the instances, and its solutions are never more than 6% worse than the best solution found. More importantly, its performance constant is never more than 23% larger than the lower bound $\mathsf{K}_{\min}$, and, on average, it is less than 7% larger than this bound. In fact, only $\Sigma_{\mathsf{fat-snake}}$ happens sometimes to find better solutions than Σ_{greedy}; however, these improvements are marginal, as one can see by comparing the absolute performance of Σ_{greedy} and *best-of*.

5.3 Choosing the Optimal Number of Chunks

To this point, we have assumed that the number n of chunks per computer was given to us. In fact, we show now that (happily) one does not have to guess at this value. We begin to flesh out this remark by noting that we can easily obtain an explicit expression for the expected work completed by any group schedule under the charged-initiation model, from that schedule's analogous expectation under the free-initiation model.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

$\mathsf{K}^{(\Sigma_{\mathsf{cyclic}})} = 34104$

1	2	3	4	5
6	7	8	9	10
15	14	13	12	11
20	19	18	17	16

$\mathsf{K}^{(\Sigma_{\mathsf{mirror}})} = 27284$

1	2	3	4	5
10	9	8	7	6
15	14	13	12	11
20	19	18	17	16

$\mathsf{K}^{(\Sigma_{\mathsf{reverse}})} = 24396$

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16

$\mathsf{K}^{(\Sigma_{\mathsf{snake}})} = 25784$

1	2	3	4	5
14	12	10	8	6
15	13	11	9	7
16	17	18	19	20

$\mathsf{K}^{(\Sigma_{\mathsf{fat-snake}})} = 24276$

1	2	3	4	5
10	9	8	7	6
15	14	13	12	11
20	19	18	16	17

$\mathsf{K}^{(\Sigma_{\mathsf{greedy}})} = 24390$

1	2	3	4	5
13	10	6	9	7
18	15	14	11	8
20	16	19	12	17

$\mathsf{K}^{(\Sigma_{\mathrm{Optimal}})} = 23780$

Table 5: Comparing group schedules for $\mathsf{n} = 20$ and $\mathsf{g} = 4$. Here the most efficient group schedules are $\Sigma_{\mathsf{fat-snake}}$, Σ_{greedy}, and $\Sigma_{\mathsf{reverse}}$ (in this order). The lower bound, $\mathsf{K}_{\min} = 23780$, is reached on this example.

Theorem 7. (p remote computers: charged-initiation model)
Let $\mathcal{C}$ be a group schedule defined by the execution chart

$$C_{i,j}\,\big|_{i\in\{1,\ldots,\mathsf{g}\},j\in\{1,\ldots,\mathsf{n}/\mathsf{g}\}}\ .$$

Then, whatever the (non-decreasing) risk function, we have:

$$E^{(c,\mathsf{n})}(\mathsf{sl}^{(c)},\mathcal{C}) \;=\; \frac{\mathsf{sl}^{(c)}}{\mathsf{sl}^{(c)}+\mathsf{n}\varepsilon}E^{(f,\mathsf{n})}(\mathsf{sl}^{(c)}+\mathsf{n}\varepsilon,\mathcal{C}).$$

Proof. To establish the result, we only need to explicit the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ under the

	Relative				Absolute				Success rate
	min	max	avg.	stdv.	min	max	avg.	stdv.	
Cyclic	1.1	3.786	2.143	0.664	1.1	3.786	2.239	0.592	00.00%
Reverse	1	1.295	1.055	0.065	1	1.295	1.117	0.061	12.42%
Mirror	1	2.468	1.504	0.393	1	2.468	1.575	0.338	12.37%
Snake	1	1.199	1.127	0.059	1	1.291	1.193	0.059	12.34%
Greedy	1	1.055	1.005	0.015	1	1.224	1.067	0.074	83.01%
Fat snake	1	1.442	1.123	0.115	1	1.530	1.192	0.143	17.07%
Best-of	1	1	1	0	1	1.224	1.061	0.069	100.00%

Table 6: Statistics on the K value of all heuristics for $2 \leq g \leq 100$ and $2g \leq n \leq 1000$ (minimum, maximum, average value and standard deviation over the 4032 instances).

charged-initiation model:

$$\begin{aligned}
E^{(c,\mathsf{n})}(\mathsf{sl}^{(c)},\mathcal{C}) &= \sum_{j=1}^{\mathsf{n}} \frac{\mathsf{sl}^{(c)}}{\mathsf{n}} \left(1 - \prod_{i=1}^{\mathsf{g}} Pr^{(c)}\left(C_{i,j}\frac{\mathsf{sl}^{(c)}}{\mathsf{n}}\right)\right) \\
&= \sum_{j=1}^{\mathsf{n}} \frac{\mathsf{sl}^{(c)}}{\mathsf{n}} \left(1 - \prod_{i=1}^{\mathsf{g}} Pr^{(f)}\left(C_{i,j}\left(\frac{\mathsf{sl}^{(c)}}{\mathsf{n}} + \varepsilon\right)\right)\right) \\
&= \frac{\mathsf{sl}^{(c)}}{\mathsf{sl}^{(c)} + \mathsf{n}\varepsilon} \sum_{j=1}^{\mathsf{n}} \left(\frac{\mathsf{sl}^{(c)}}{\mathsf{n}} + \varepsilon\right)\left(1 - \prod_{i=1}^{\mathsf{g}} Pr^{(f)}\left(C_{i,j}\left(\frac{\mathsf{sl}^{(c)}}{\mathsf{n}} + \varepsilon\right)\right)\right) \\
&= \frac{\mathsf{sl}^{(c)}}{\mathsf{sl}^{(c)} + \mathsf{n}\varepsilon} E^{(f,\mathsf{n})}(\mathsf{sl}^{(c)} + \mathsf{n}\varepsilon, \mathcal{C}).
\end{aligned}$$

□

Now we can determine the value of n, making only the assumption that the expectation of the group schedule within the charged-initiation model is a unimodal function of n. (It is quite natural to assume that this expectation is non-decreasing with n under the free-initiation model.) We can, then, use a binary search to seek the optimum value of n. Specifically, for each tested value m we compare the values of the expectation for m and $m+1$ to determine if the expectation is still increasing in m, in which case m is smaller than the optimum n. The binary search can be safely performed in the interval $[1..X/\varepsilon]$.

6 Experiments

We have performed a suite of simulation experiments in order to gain insight into the performance of the group heuristics on simulated platforms that are subject to unrecoverable interruptions. We report only on the observed behavior of Σ_{greedy} for two reasons, first because of its preeminence in the experiment reported in Table 6 and, second, because our simulations show only small differences among our six heuristics. The source code for all six group heuristics can be found at http://graal.ens-lyon.fr/~abenoit/code/failure.c.

6.1 The Experimental Plan

We use randomly generated platforms made of p computers. In all experiments, we set $\kappa = 1$, and we choose the times for interruptions randomly between 0 and 1, following a uniform distribution. The size of the workload, $W_{(\mathrm{ttl})}$, varies between 1 and p. $W_{(\mathrm{ttl})} = 1$ represents the case in which all computers can potentially do all the work before being interrupted; $W_{(\mathrm{ttl})} = p$ represents the case in which we can do no better than deploy one different slice of size 1 on each computer (which will then compute until it is interrupted), using no replication at all.

The key parameters in our experiment are: the number of computers, p; the total amount of work, $W_{(\mathrm{ttl})}$; the number of chunks per unit of work, n; and the start-up cost, ε. In the first four series of experiments, three of these parameters are fixed while the fourth one varies. When fixed, these parameters take the following values:

- $p = 5, 10, 25, 50$, or 100;
- $W_{(\mathrm{ttl})} = 0.3p$ or $0.7p$;
- $n = 47, 97, 147$, or 197;
- $\varepsilon = 0.1000, 0.0100, 0.0010$, or 0.0001.

These parameters are defined over large ranges of values in order to assess the heuristics in very different configurations, even very unfavorable ones.

We compare several heuristics:

Σ_{brute}– This *brute replication heuristic* replicates the entire workload onto all computers. Each computer executes work in the order of receipt, starting from the first chunk, until it is interrupted.

$\Sigma_{\text{no-rep}}$– This *no replication heuristic* distributes the work in a round-robin fashion, with no replication. Thus, each computer is allocated $W_{(\text{ttl})}/p$ units of work (rounded by the chunk size).

$\Sigma_{\text{cyclic-rep}}$– This *cyclic replication heuristic* distributes the work in a round-robin fashion, as does $\Sigma_{\text{no-rep}}$, but it keeps distributing chunks, starting from chunk 1 again, until each computer has a total (local) workload of 1. Note that when the number of chunks is a multiple of p, this heuristic is identical to $\Sigma_{\text{no-rep}}$, since the chunks assigned to a computer during the replication phase were already assigned to it previously.

$\Sigma_{\text{random-rep}}$– This *random replication heuristic* distributes a total workload of 1 to each computer, but it chooses the chunks and their order randomly, while ensuring that all chunks deployed on the same computer are distinct. However, the same chunk can be assigned to several computers.

Σ_{greedy}– This *group greedy heuristic* is the schedule Σ_{greedy} of Section 5.2.2. Since our number of chunks n may not be a multiple of g, the last group of computers may not have a full g chunks to process. The scheduling heuristic works *as if* the last group contained g chunks, hence potentially inserting idle time-slots in the schedule. During the schedule execution these idle time-slots are obviously skipped (a computer is not kept idle when it still has work to process).

The values for n where picked so has not to favor the group-heuristics by almost certainly ensuring that the last group of computers never has a full g chunks to process.

$\Sigma_{\text{omniscient}}$– This last *omniscient heuristic* is an idealized static heuristic that knows exactly when each computer is interrupted. This idealized knowledge obviates replication: each computer is statically allocated a single chunk whose length, plus the length of the start-up cost ε is exactly equal to the time before failure of the computer. Therefore, this heuristic returns the maximal work that could be done, knowing the failure times.

We do not report the absolute amount of work done by the heuristics as this would be meaningless, as the amount of work distributed, and the amount of work that can be processed before all computers fail, both vary vastly between experiments. We therefore consider, on each instance and for each heuristic, the ratio between the work completed by that heuristic and the work completed by $\Sigma_{\text{omniscient}}$. With our measure, $\Sigma_{\text{omniscient}}$ always achieves a performance of 1 and we do not display it on figures. (In the cases where $\Sigma_{\text{omniscient}}$ does not complete any work—namely, cases where all computers fail at times smaller than the start-up costs—the performance of all heuristics is set to 1.)

6.2 Experimental Results

For each considered set of parameters, 100 different failure-configurations were randomly built (computer failure times). We report the average of these results.

6.2.1 Experiment (E1): Fixed p, n, and ε

In this first experiment, we analyse the impact of the workload on the heuristics. The total amount of work $W_{(\text{ttl})}$ varies between 1 and the number of computers p, which are the two extreme cases. The other parameters are fixed.

Figure 7 presents some representative results. (All graphs are presented in the appendix on Figures 14 through 23.)

When $W_{(\text{ttl})} = 1$, opportunities for replication are maximum. As anyone could have foreseen, in this case $\Sigma_{\text{random-rep}}$ often dominates $\Sigma_{\text{no-rep}}$. Replication is therefore worth considering in the general case. Replication, however, should be done in a meaningful way: Σ_{brute} almost always achieve very poor performance.

Another obvious conclusion is that when $W_{(\text{ttl})} = p$, there is no room for replication and $\Sigma_{\text{no-rep}}$ is equivalent to $\Sigma_{\text{cyclic-rep}}$ and Σ_{greedy}.

In all cases, $\Sigma_{\text{cyclic-rep}}$ achieves better performance than $\Sigma_{\text{no-rep}}$. This is significant when $W_{(\text{ttl})}$ is small with respect to pX. These two heuristics are equivalent when the total number of chunks is a multiple of the number of computers. On each instance the best performance is always achieved by Σ_{greedy}.

When there is very little room for replication, i.e., $W_{(\text{ttl})}$ is close to p, $\Sigma_{\text{no-rep}}$, $\Sigma_{\text{cyclic-rep}}$, and Σ_{greedy} achieve similar performance.

This experiment was not supposed to focus on the influence of ε or of the number of chunks. However, one can easily see that the performance is always bad when the number of chunks is too large considering the start-up costs (for instance, when $\varepsilon = 0.01$, with 47 chunks the start-up cost accounts for roughly a third of the size of each chunk).

6.2.2 Experiment (E2): Fixed $W_{(\text{ttl})}$, cs, and ε

In this second experiment, we study the behavior of the heuristics when the number of computers varies, from 1 up to 100. For the comparison to be fair and different from (E1), the total amount of work $W_{(\text{ttl})}$ is always kept proportional to the number p of computers (either $W_{(\text{ttl})} = 0.3p$ or $W_{(\text{ttl})} = 0.7p$).

Figure 8 presents a representative excerpt of the experiments. (All results are presented in the appendix, on Figures 24 through 27).

Heuristic Σ_{brute} sees its relative performance dramatically drop when the number of computers grows. Otherwise, the only other heuristic impacted by the number of computers is $\Sigma_{\text{cyclic-rep}}$ whose performance increases when, roughly, there are more than 10 computers. Otherwise, the conclusions are mainly the same than for experiments (E1): when $\frac{W_{(\text{ttl})}}{p}$ is small Σ_{greedy} and $\Sigma_{\text{cyclic-rep}}$ outperform $\Sigma_{\text{no-rep}}$; when $\frac{W_{(\text{ttl})}}{p}$ is large the three heuristics have similar performance but Σ_{greedy} is always the best heuristic; $\Sigma_{\text{random-rep}}$ has a significantly lower performance. A new conclusion is that, when $\frac{W_{(\text{ttl})}}{p}$ is larger, there is less room for replication, efficient use of resources is more complicated, and the heuristics have overall worse performance.

The main conclusion of this experiment is that the performance of the heuristics scale very well to large platforms.

6.2.3 Experiment (E3): Fixed $W_{(\text{ttl})}$, p, and ε

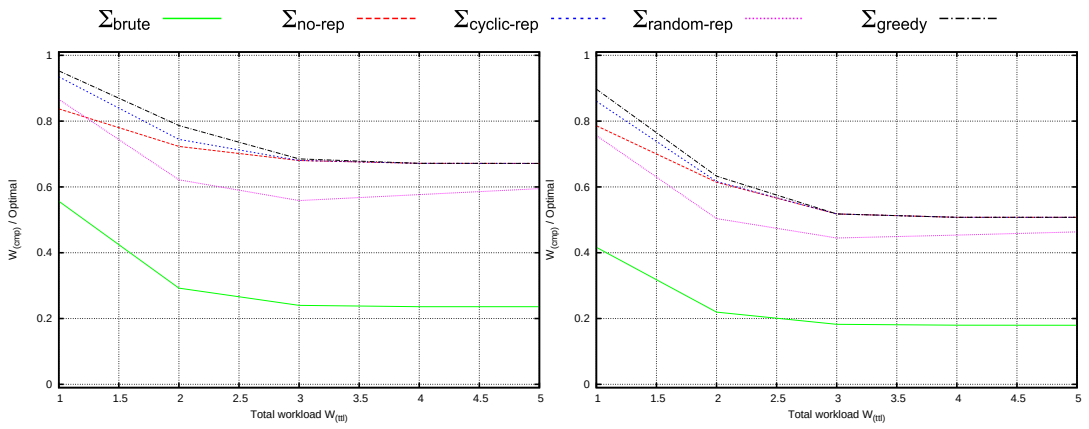
In this experiment the only varying parameter is the number of chunks per unit of work, which takes any odd values less than 200.

Figure 9 presents a representative excerpt of the experiments. (All results are presented in the appendix, on Figures 28 through 32).

When the start-up cost is negligible, one should use a large number of chunks, since having small chunks reduces the loss occurred when a computer fails. However, when the start-up cost increases, one should be more cautious because the start-up cost then impacts negatively the performance of the solution. For large start-up costs, the decrease of performance is dramatic. This is less obvious in the intermediate case of $\varepsilon = 0.001$ but, even in this case, after reaching a maximum, the performance decreases when the number of chunks increases. The general shape of the curves corroborate the unimodal assumption proposed at the end of Section 5.3.

Of course, special care should be taken about the exact number of chunks if using $\Sigma_{\text{cyclic-rep}}$ whose performance fluctuates, depending on whether the number of computers is prime with the number of chunks.

As the studied parameter is not the overall number of chunks but the number of chunks per unit of work, the number of computers has no significant impact on the performance (except, obviously, for $\Sigma_{\text{cyclic-rep}}$).



(a) 5 computers, $\varepsilon = 0.0100$, 47 chunks.

(b) 5 computers, $\varepsilon = 0.0100$, 97 chunks.

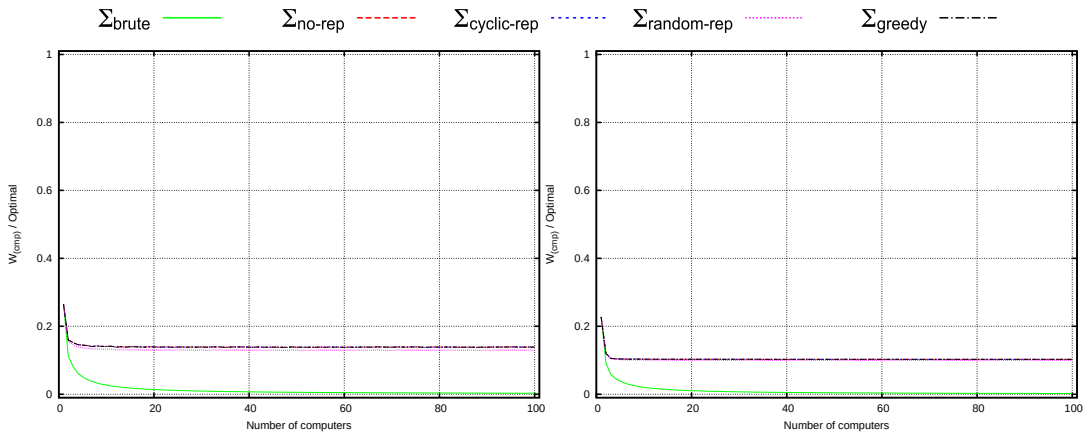
(c) 25 computers, $\varepsilon = 0.0010$, 147 chunks.

(d) 25 computers, $\varepsilon = 0.0001$, 197 chunks.

(e) 100 computers, $\varepsilon = 0.0001$, 147 chunks.

(f) 100 computers, $\varepsilon = 0.0001$, 197 chunks.

Figure 7: (E1): representative sampling of studied configurations.



(a) $W_{(\mathrm{ttl})} = 30$, $\varepsilon = 0.1000$, and 97 chunks.

(b) $W_{(\mathrm{ttl})} = 70$, $\varepsilon = 0.1000$, and 97 chunks.

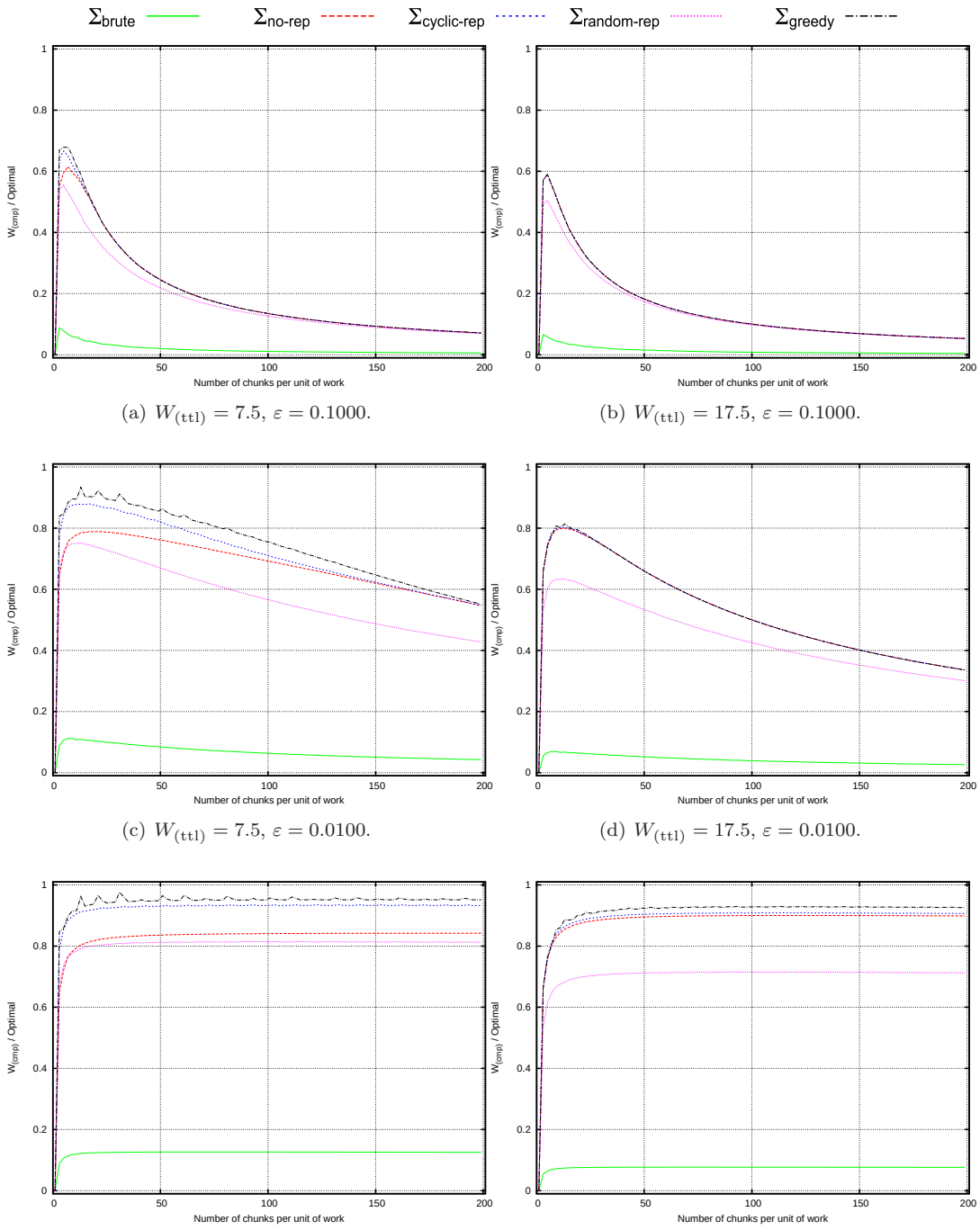
(c) $W_{(\mathrm{ttl})} = 30$, $\varepsilon = 0.0100$, and 97 chunks.

(d) $W_{(\mathrm{ttl})} = 70$, $\varepsilon = 0.0100$, and 97 chunks.

(e) $W_{(\mathrm{ttl})} = 30$, $\varepsilon = 0.0010$, and 97 chunks.

(f) $W_{(\mathrm{ttl})} = 70$, $\varepsilon = 0.0010$, and 97 chunks.

Figure 8: (E2): representative sampling of studied configurations ($p = 100$).



(a) $W_{(\mathrm{ttl})} = 7.5$, $\varepsilon = 0.1000$.

(b) $W_{(\mathrm{ttl})} = 17.5$, $\varepsilon = 0.1000$.

(c) $W_{(\mathrm{ttl})} = 7.5$, $\varepsilon = 0.0100$.

(d) $W_{(\mathrm{ttl})} = 17.5$, $\varepsilon = 0.0100$.

(e) $W_{(\mathrm{ttl})} = 7.5$, $\varepsilon = 0.0001$.

(f) $W_{(\mathrm{ttl})} = 17.5$, $\varepsilon = 0.0001$.

Figure 9: (E3): representative sampling of studied configurations ($p = 25$).

6.2.4 Experiment (E4): Fixed $W_{(\text{ttl})}$, p, and cs

In this set of experiments, we study the impact of the start-up cost on the solution, which can take values between 0 and 1.

Figure 10 presents a representative excerpt of the experiments. (All results are presented in the appendix, on Figures 33 through 37).

When the start-up cost ε increases, starting from 0, one observes a dramatic drop in performance. Indeed, when ε is large, that is, when $\varepsilon \geq 0.05$ (roughly), very few chunks can be executed on a computer before it fails. In these configurations, the performance mainly depends on the size of chunks with respect to the failure times in the instance. There is no way to design good heuristics on average (compared to $\Sigma_{\mathsf{omniscient}}$) and all heuristics have poor performance. This even gets worse with the increase of the number of chunks per unit of work. As ε gets closer to 1, the proportion of cases where even $\Sigma_{\mathsf{omniscient}}$ do not complete any work increases. In these cases, all heuristics have a performance of 1, hence the sharp increase in heuristic performance. Needless to say that these cases have no practical merits.

6.2.5 Experiment (E5) and (E6): Automatic Inference of Chunk Size

Finally, we did two different set of experiments to assess the quality of the heuristics when the number of chunks is automatically inferred using the scheme proposed in Section 5.3.

Experiment (E5) replicates Experiment (E1) except that, for each instance and each heuristic, the chunk size is no longer given but automatically inferred. Figure 11 presents an aggregated view of the 76,000 generated instances. (All results are presented in the appendix, on Figures 62 and 63). On Figure 11, for each heuristic we plot the average performance when only considering the x% best instances for that heuristic. The performance for 100% is thus the average performance over all instances: 85.2% of the omniscient optimal for Σ_{greedy} and 79.7% for $\Sigma_{\mathsf{no\text{-}rep}}$. Therefore, on average, Σ_{greedy} closed 37.8% of the gap between $\Sigma_{\mathsf{no\text{-}rep}}$ and the optimal. Furthermore, in more than 21% of the instances Σ_{greedy} achieves quasi-optimal performance (over 99.5%). $\Sigma_{\mathsf{cyclic\text{-}rep}}$ achieves close performance.

In Experiment (E6), presented on Figure 12, we fixed the overall workload to 10 units ($W_{(\text{ttl})} = 10$) and we had the number of computers take any integral value between 10 and 100 (with the same four choices for the value of ε as previously). This scheme enables to assess the impact of the ratio of potential replication, $\frac{pX}{W_{(\text{ttl})}}$. We randomly built 1000 instances of each set of parameters.

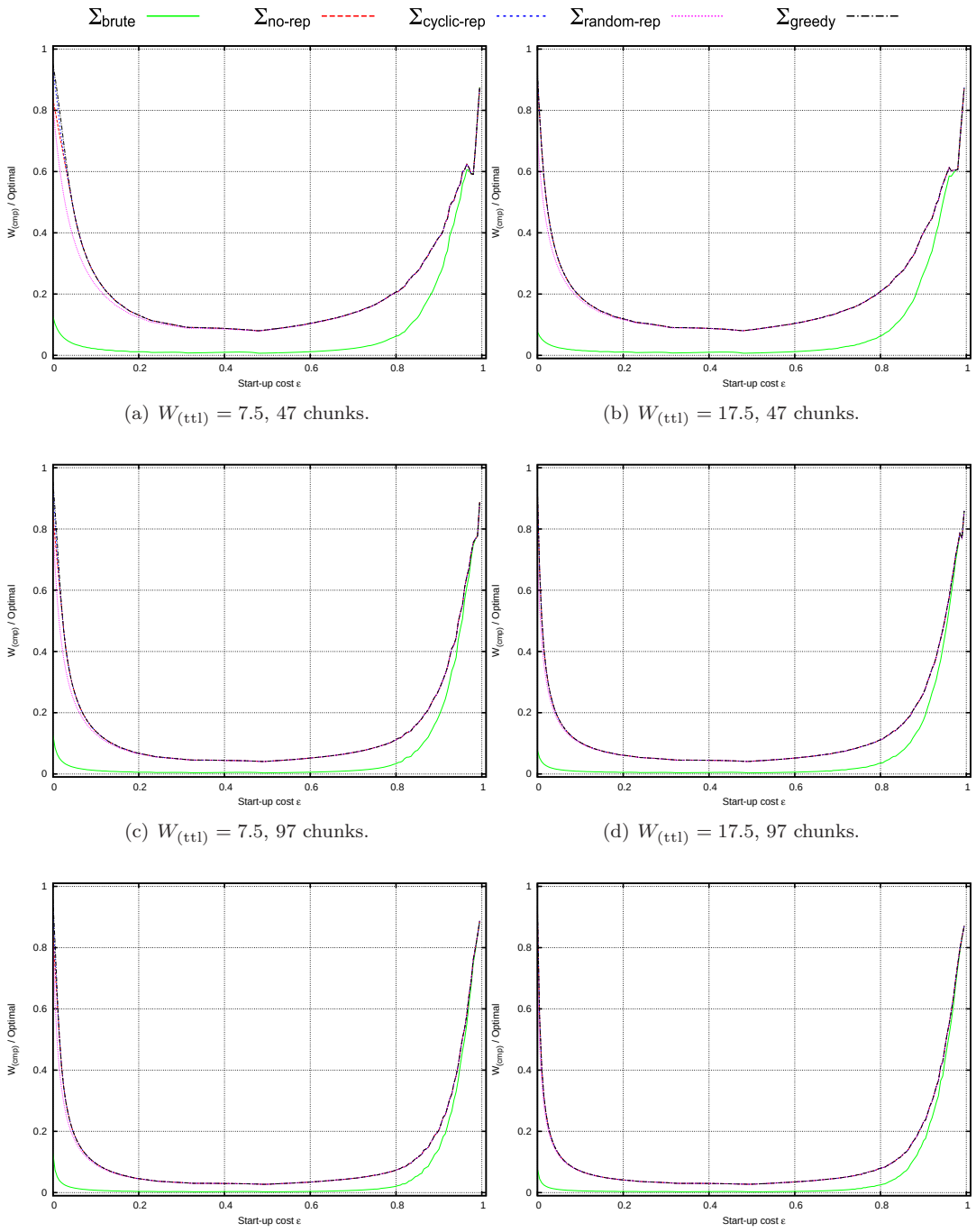
$\Sigma_{\mathsf{cyclic\text{-}rep}}$ and Σ_{greedy} always have better performance than $\Sigma_{\mathsf{no\text{-}rep}}$, and the difference is very significant as soon as the ratio of potential replication is greater than 2. Σ_{greedy} has better and more regular performance than $\Sigma_{\mathsf{cyclic\text{-}rep}}$. $\Sigma_{\mathsf{cyclic\text{-}rep}}$ takes almost no advantage of the possibility of replication when the potential for replication is small (smaller than 2); Σ_{greedy} takes advantage.

6.3 Summarizing the Experiments

From these experiments, we see that replication, when cleverly done, can improve the performance of heuristics. Our Σ_{greedy} heuristic always delivers good performance, is never outperformed by any other heuristic (on each configuration, on average, it delivers the best performance), and, on favorable cases, it performs significantly better than any other heuristics. This heuristic is therefore an obvious winner.

7 Going Beyond the Linear Risk Model

So far, we have almost exclusively focused on the linear risk model. Some of our results, however, can be extended to general risk functions (we have directly written Theorems 4 and 7 in a general context). We will first extend two of our results to the general case (Section 7.1). Inferring from these theoretical results, we will extend our group heuristics to the general context and evaluate them (Section 7.2).



(a) $W_{(\mathrm{ttl})} = 7.5$, 47 chunks.

(b) $W_{(\mathrm{ttl})} = 17.5$, 47 chunks.

(c) $W_{(\mathrm{ttl})} = 7.5$, 97 chunks.

(d) $W_{(\mathrm{ttl})} = 17.5$, 97 chunks.

(e) $W_{(\mathrm{ttl})} = 7.5$, 147 chunks.

(f) $W_{(\mathrm{ttl})} = 17.5$, 147 chunks.

Figure 10: (E4): representative sampling of studied configurations ($p = 25$).

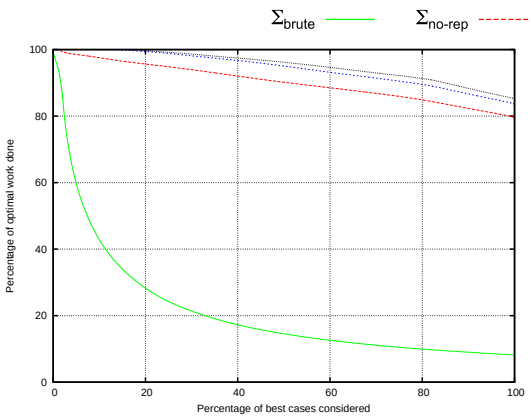


Figure 11: Experiment (E5): performance with automatic inference of chunk sizes.

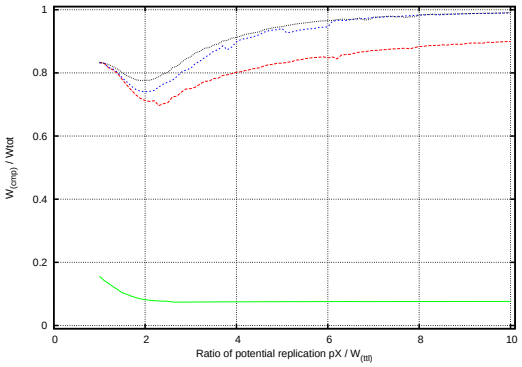


Figure 12: Experiment (E6): impact of the ratio of potential replication, $\frac{pX}{W_{(\mathrm{ttl})}}$.

7.1 Asymptotically Optimal Scheduling under General Risk and the Free-Initiation Model

We prove that, with one or two remote computers, a schedule using equal-size chunks is asymptotically optimal.

7.1.1 One Remote Computer

Theorem 8. ***(One remote computer: free-initiation model and general risk)*** *Say that one wishes to deploy $W_{(\mathrm{ttl})}$ units of work to a single remote computer in at most n chunks, for some strictly positive integer n. The scheduling regimen below, which partitions the overall workload into n equal-size chunks, is asymptotically optimal:*

$$\forall i \in [1,n]\,, \mathcal{W}_i \leftarrow \left\langle \frac{i-1}{n} W_{(\mathrm{ttl})},\ \frac{i}{n} W_{(\mathrm{ttl})} \right\rangle.$$

In other words, the expectation of $\mathsf{W}_{(\mathrm{cmp})}$ of this equal-size regimen tends to the expectation of an optimal regimen as n grows without bound.

Note that, if there exists a minimal amount of work V by which the computer is certain to be interrupted (with probability 1), then one can improve the regimen with equal-size chunks by having chunks of size $\frac{\min\{W_{(\mathrm{ttl})},V\}}{n}$ rather than of size $\frac{W_{(\mathrm{ttl})}}{n}$. (Under the linear-risk model $V = X$.)

Proof. In this proof, we denote by $\mathcal{S}(n)$ the regimen using n equal-size chunks we want to establish the asymptotic optimality of. We denote by $\mathcal{O}(n)$ an optimal regimen using (at most) n chunks. Under regimen $\mathcal{O}(n)$, we denote the chunks $\mathcal{W}'_1$, ..., $\mathcal{W}'_n$, and we denote by ω'_i the size of chunk $\mathcal{W}'_i$. Without loss of generality, we can assume that, for any i in $[1..n]$, chunk $\mathcal{W}'_i$ is equal to $\left\langle \sum_{k=1}^{i-1} \omega'_k, \sum_{k=1}^{i} \omega'_k \right\rangle$.

Let us consider any strictly positive integer m. We are going to compare the performance of the scheduling regimens $\mathcal{O}(n)$ and $\mathcal{S}(m)$. For that purpose, we introduce three more notations. First, we denote by α the size of a chunk of $\mathcal{S}(m)$: $\alpha = \frac{W_{(\mathrm{ttl})}}{m}$. Then, for any $i \in [1..m-1]$, let $s(i)$ be the index of the first chunk of $\mathcal{S}(m)$ which starts no sooner than the end of the i-th chunk of $\mathcal{O}(n)$. Formally:

$$s(i) = 1 + \left\lceil \frac{\sum_{k=1}^{i} \omega'_k}{\alpha} \right\rceil.$$

Symmetrically, for any $i \in [1..m]$, let $p(i)$ be the index of the last chunk of $\mathcal{S}(m)$ which ends no later than the beginning of the i-th chunk of $\mathcal{O}(n)$. Formally:

$$p(i) = \left\lfloor \frac{\sum_{k=1}^{i-1} \omega'_k}{\alpha} \right\rfloor .$$

If at least one chunk of $\mathcal{S}(m)$ is fully included in $\mathcal{W}'_i$, $s(i-1)$ is the index of the first such chunk, and $p(i+1)$ the index of the last such chunk.

The overall expectation of $\mathsf{W}_{(\mathrm{cmp})}$ for $\mathcal{O}(n)$ is:

$$E(W_{(\mathrm{ttl})}, \mathcal{O}(n)) = \sum_{i=1}^{n} \omega'_i \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right). \tag{19}$$

Let us now consider any chunk $\mathcal{W}'_i$ of $\mathcal{O}(n)$ (that is, any $i \in [1..n]$). Its contribution to the overall expectation is:

$$e_i = \omega'_i \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right). \tag{20}$$

If $\omega'_i < 2\alpha$, obviously $e_i < 2\alpha$. Otherwise, $\omega'_i \geq 2\alpha$ and there exists at least one chunk of $\mathcal{S}(m)$ which is included in the chunk $\mathcal{W}_i$. Then, $p(i+1) \geq s(i-1)$ (we extend s by letting $s(0) = 0$). We can then establish:

$$\begin{aligned}
\omega'_i &= \sum_{j=1}^{i} \omega'_j - \sum_{j=1}^{i-1} \omega'_j \\
&= \left(\sum_{j=1}^{i} \omega'_j - p(i+1)\alpha\right) + (p(i+1)\alpha - (s(i-1)-1)\alpha) \\
&\quad + \left((s(i-1)-1)\alpha - \sum_{j=1}^{i-1} \omega'_j\right) \\
&< \alpha + (p(i+1) - s(i-1) + 1)\,\alpha + \alpha.
\end{aligned}$$

Using this result and Equation (20) we can bound the value of e_i:

$$\begin{aligned}
e_i &< (2\alpha + (p(i+1) - s(i-1) + 1)\,\alpha)\left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right) \\
&\leq 2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_j\right)\right) \\
&\leq 2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha\,(1 - Pr\,(j\alpha))\,.
\end{aligned}$$

The last inequation holds because $p(i+1)\alpha$ is no greater than $\sum_{j=1}^{i} \omega'_j$ and because Pr is a non decreasing function.

We can now rewrite Equation (19):

$$\begin{aligned} E(W_{(\text{ttl})}, \mathcal{O}(n)) &= \sum_{\substack{1\leq i\leq n \\ \omega_i'<2\alpha}} e_i + \sum_{\substack{1\leq i\leq n \\ \omega_i'\geq 2\alpha}} e_i \\ &< \sum_{\substack{1\leq i\leq n \\ \omega_i'<2\alpha}} 2\alpha + \sum_{\substack{1\leq i\leq n \\ \omega_i'\geq 2\alpha}} \left(2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha\left(1 - Pr\left(j\alpha\right)\right)\right) \\ &\leq 2n\alpha + \sum_{j=1}^{m} \alpha\left(1 - Pr\left(j\alpha\right)\right) \\ &\leq \frac{2n}{m} W_{(\text{ttl})} + E(W_{(\text{ttl})}, \mathcal{S}(m)). \end{aligned}$$

Therefore, for any positive integers n and m:

$$E(W_{(\text{ttl})}, \mathcal{O}(n)) - \frac{2n}{m} W_{(\text{ttl})} < E(W_{(\text{ttl})}, \mathcal{S}(m)) \leq E(W_{(\text{ttl})}, \mathcal{O}(m)). \tag{21}$$

$E(W_{(\text{ttl})}, \mathcal{O}(n))$ is obviously a non-decreasing, upper-bounded (by $W_{(\text{ttl})}$), sequence and it thus converges. By replacing n by $\lfloor\sqrt{m}\rfloor$ in Equation (21), one easily sees that the sequence $E(W_{(\text{ttl})}, \mathcal{S}(m))$ is converging with the same limit.

□

7.1.2 Two Remote Computers

Theorem 9. ***(Two remote computers: free-initiation model and general risk)*** *Say that one wishes to deploy $W_{(\text{ttl})}$ units of work on two computers, in at most n chunks, for some strictly positive integer n. Then, the following regimen, which schedules the same set of equal-size chunks on both computers, is asymptotically optimal:*

$$\forall i \in [1, n], \mathcal{W}_{1,i} = \mathcal{W}_{2,n-i+1} \leftarrow \left\langle \frac{i-1}{n} W_{(\text{ttl})},\ \frac{i}{n} W_{(\text{ttl})} \right\rangle.$$

In other words, the expectation of $\mathsf{W}_{(\text{cmp})}$ of the above regimen tends to the expectation of an optimal regimen as n grows without bound.

Proof. This proof is mainly a combination of the result of Theorem 4 and of the proofs of Theorem 6 and 8.

In this proof, we denote by $\mathcal{S}(n)$ the regimen using n chunks we want to establish the asymptotic optimality of. We denote by $\mathcal{O}(n)$ an optimal regimen using (at most) n chunks.

Thanks to Theorem 4, we know the general shape of schedule $\mathcal{O}(n)$. Without loss of generality, we can indeed assume that $\mathcal{O}(n)$ has the shape described on Figure 4(a). We can then use the first two transformations of Figure 4. We first complete the workload of each computer (Figure 4(b)) and then subdivide the chunks for chunk boundaries to coincide (Figure 4(c)). This way we transform $\mathcal{O}(n)$ into a scheduling regimen $\mathcal{O}'(n)$ with at most $l = 2n + 1$ chunks per computer and whose expectation is no smaller than that of $\mathcal{O}$ (following the arguments already used in the proof of Theorem 6):

$$E(W_{(\text{ttl})}, \mathcal{O}(n)) \leq E(W_{(\text{ttl})}, \mathcal{O}'(n)).$$

Under regimen $\mathcal{O}'(n)$, we denote by $\mathcal{W}'_{1,1}$, ..., $\mathcal{W}'_{1,l}$ the chunks of computer P_1 and by $\mathcal{W}'_{2,1}$, ..., $\mathcal{W}'_{2,l}$ those of computer P_2. Then, for any $i \in [1..l]$, $\mathcal{W}'_{2,l-i+1} = \mathcal{W}'_{1,i}$.

Let us now consider any strictly positive integer m. We are going to compare the performance of the scheduling regimens $\mathcal{O}'(n)$ and $\mathcal{S}(m)$.

For that purpose, we introduce three more notations. First, we denote by α the size of a chunk of $\mathcal{S}(m)$: $\alpha = \frac{W_{(\text{ttl})}}{m}$. Then, for any $i \in [1..m-1]$, let $s(i)$ be the index of the first chunk of $\mathcal{S}(m)$

which starts no sooner than the end of the i-th chunk of $\mathcal{O}'(n)$ (on computer P_1). Formally:

$$s(i) = 1 + \left\lceil \frac{\sum_{k=1}^{i} \omega'_{1,i}}{\alpha} \right\rceil .$$

Symmetrically, for any $i \in [1..m]$, let $p(i)$ be the index of the last chunk of $\mathcal{S}(m)$ which ends no later than the beginning of the i-th chunk of $\mathcal{O}'(n)$ (on computer P_1). Formally:

$$p(i) = \left\lfloor \frac{\sum_{k=1}^{i-1} \omega'_{1,i}}{\alpha} \right\rfloor .$$

The overall expectation of $\mathsf{W}_{(\mathrm{cmp})}$ for $\mathcal{O}'(n)$ is:

$$E(W_{(\mathrm{ttl})}, \mathcal{O}'(n)) = \sum_{i=1}^{l} \omega'_{1,i} \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1} \omega'_{1,j}\right)\right). \tag{22}$$

Let us now consider any chunk $\mathcal{W}'_{1,i}$ of $\mathcal{O}'(n)$ (that is, any $i \in [1..m]$). Its contribution to the overall expectation is:

$$e_i = \omega'_{1,i} \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1} \omega'_{1,j}\right)\right). \tag{23}$$

If $\omega'_{1,i} < 2\alpha$, obviously $e_i < 2\alpha$. Otherwise, $\omega'_{1,i} \geq 2\alpha$ and there exists at least one chunk of $\mathcal{S}(m)$ which is included in the chunk $\mathcal{W}_{1,i}$. Then, $p(i+1) \geq s(i-1)$ (we extend s by letting $s(0) = 0$). We can then establish:

$$\begin{aligned}
\omega'_{1,i} &= \sum_{j=1}^{i} \omega'_{1,j} - \sum_{j=1}^{i-1} \omega'_{1,j} \\
&= \left(\sum_{j=1}^{i} \omega'_{1,j} - p(i+1)\alpha\right) + (p(i+1)\alpha - (s(i-1)-1)\alpha) \\
&\quad + \left((s(i-1)-1)\alpha - \sum_{j=1}^{i-1} \omega'_{1,j}\right) \\
&< \alpha + (p(i+1) - s(i-1) + 1)\,\alpha + \alpha.
\end{aligned}$$

Using this result and Equation (23) we can bound the value of e_i:

$$\begin{aligned}
e_i &< (2\alpha + (p(i+1) - s(i-1) + 1)\,\alpha) \\
&\qquad \times \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1} \omega'_{1,j}\right)\right) \\
&\leq 2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha \left(1 - Pr\left(\sum_{j=1}^{i} \omega'_{1,j}\right) Pr\left(W_{(\mathrm{ttl})} - \sum_{j=1}^{i-1} \omega'_{1,j}\right)\right) \\
&\leq 2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha \left(1 - Pr\,(j\alpha)\, Pr\left(W_{(\mathrm{ttl})} - (j-1)\alpha\right)\right).
\end{aligned}$$

The last inequation holds because Pr is a non decreasing function, because $p(i+1)\alpha$ is no greater than $\sum_{j=1}^{i} \omega'_{1,j}$, and because $(s(i-1)-1)\alpha$ is no smaller than $\sum_{j=1}^{i-1} \omega'_{1,j}$.

We can now rewrite Equation (22):

$$\begin{aligned} E(W_{(\text{ttl})}, \mathcal{O}'(n)) &= \sum_{\substack{1\le i\le l \\ \omega'_{1,i}<2\alpha}} e_i + \sum_{\substack{1\le i\le l \\ \omega'_{1,i}\ge 2\alpha}} e_i \\ &< \sum_{\substack{1\le i\le l \\ \omega'_{1,i}<2\alpha}} 2\alpha \\ &\quad + \sum_{\substack{1\le i\le l \\ \omega'_{1,i}\ge 2\alpha}} \left(2\alpha + \sum_{j=s(i-1)}^{p(i+1)} \alpha \left(1 - Pr\left(j\alpha\right) Pr\left(W_{(\text{ttl})} - (j-1)\alpha\right)\right)\right) \\ &\le 2l\alpha + \sum_{j=1}^{l} \alpha \left(1 - Pr\left(j\alpha\right) Pr\left(W_{(\text{ttl})} - (j-1)\alpha\right)\right) \\ &\le \frac{4n+2}{m} W_{(\text{ttl})} + E(W_{(\text{ttl})}, \mathcal{S}(m)). \end{aligned}$$

Therefore, for any positive integers n and m:

$$\begin{aligned} E(W_{(\text{ttl})}, \mathcal{O}(n)) - \frac{4n+2}{m} W_{(\text{ttl})} \le E(W_{(\text{ttl})}, \mathcal{O}'(n)) - \frac{4n+2}{m} W_{(\text{ttl})} \\ < E(W_{(\text{ttl})}, \mathcal{S}(m)) \le E(W_{(\text{ttl})}, \mathcal{O}(m)). \quad (24) \end{aligned}$$

$E(W_{(\text{ttl})}, \mathcal{O}(n))$ is obviously a non-decreasing, upper-bounded (by $W_{(\text{ttl})}$), sequence and it thus converges. By replacing n by $\lfloor \sqrt{m} \rfloor$ in Equation (24), one easily sees that the sequence $E(W_{(\text{ttl})}, \mathcal{S}(m))$ is converging with the same limit. □

7.2 Heuristics and Simulations

We have shown that schedules using equal-size chunks were asymptotically optimal on systems with one or two remote computers. It is then natural to extend our group heuristics, defined in Section 5.2, for the case with general risk functions. This extension is straightforward as we only need to replace, for the group-greedy heuristic, the linear probability function with the function we wish to study. (Obviously, we also need to use the correct risk function when inferring the adequate number of chunks using the scheme of Section 5.3.)

To assess the quality of our heuristics in the general context, we use traces.

7.2.1 Traces and Methodology

We evaluate our algorithms using 8 traces recording, per computer, the lengths of the different time intervals during which the computer was available:

0. the *SDSC trace*, described in [23, p. 33], contains 5678 availability durations from a desktop grid of PC's located at the San Diego Super Computer Center (SDSC);
1. the *UCB trace*, described in [4], contains 19276 availability durations from 53 DEC workstations from the University of California, Berkeley;
2. the *XtremWeb trace*, described in [23, p. 33], contains 8756 availability durations from a desktop grid including a cluster and some PC's located at the University of Paris South;
3. the *Cetus trace*, described in [30], contains 1898 availability durations from 31 Sun workstations from the University of Tennessee;
4. the *LONG trace*, described in [30], contains 10958 availability durations from workstations located at different sites;
5. the *Princeton trace*, described in [30], contains 79 availability durations from 16 Dec Alpha workstations from the Princeton university;
6. the *Condor trace*, described in [28], contains 1125 availability durations from the Condor pool at the University of Wisconsin;

7. the *CSIL trace*, described in [28], contains 927 availability durations from the CSIL computer science student lab at the University of California, Santa Barbara.

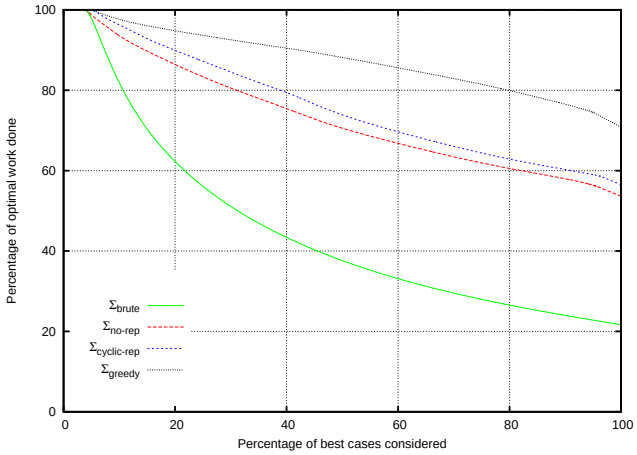
We first normalize these traces so that for each trace the longest availability interval is exactly equal to 1 (this only matters when we want to average statistics over different traces). Then, from these traces we build failure probability functions as follows:

$$Pr(trace, t) = \frac{\text{Number of availability durations in } trace \text{ that are shorter than } t}{\text{Number of availability durations in } trace}.$$

We generate instances of computer failure times by uniformly and randomly picking availability durations in the studied trace. Therefore, we implicitly assume that, when making a scheduling decision, we only consider computers that just became available.

7.2.2 Simulation Results

We ran the heuristics setting parameter λ to 1.00 (see Section 5.1), parameters p and ε according to Section 6.1, and parameter $W_{(\text{ttl})}$ taking all integral values in $[1..p]$. The aggregated simulation results are presented on Figure 13. (Ventilated graphs are presented in the appendix on Figures 64 through 67.) Overall, and under each studied scenario, Σ_{greedy} achieves far better results than $\Sigma_{\text{cyclic-rep}}$ and $\Sigma_{\text{no-rep}}$. The difference between Σ_{greedy} and the other heuristics becomes more and more important as the number of computers increases or as the size of the start-up cost decreases. The performance of $\Sigma_{\text{cyclic-rep}}$ is close to that of $\Sigma_{\text{no-rep}}$.



(a) For each of the four heuristics, the curve $y = f(x)$ shows the average performance y of the heuristics when only considering the $x\%$ cases most favorable to that heuristics.

Heuristic	average	stdv
Σ_{brute}	21.7	24.2
$\Sigma_{\text{no-rep}}$	53.6	22.1
$\Sigma_{\text{cyclic-rep}}$	56.5	22.2
Σ_{greedy}	70.8	23.2

(b) Statistics over all 608000 instances.

Figure 13: Performance of the heuristics with risk functions defined by computer-availability traces.

8 Conclusion

We have presented a model for studying the problem of scheduling large divisible workloads on p identical remote computers that are vulnerable (with the same risk function) to unrecoverable interruptions (Section 2). Our goal has been to find schedules for allocating work to the computers and for scheduling the checkpointing of that work, in a manner that maximizes the expected amount of work completed by the remote computers. Most of the results we report assume that the risk of a remote computer's being interrupted increases *linearly* with the amount of time that

the computer has been available to us; a few results provide scheduling guidelines for more general risks.

We have completely solved this scheduling problem for the case of $p = 1$ remote computer (Section 3). Our solution provides exactly optimal schedules—whose expected work completion is exactly maximum—both for the free-initiation model, wherein checkpointing incurs no overhead, and the charged-initiation model, wherein checkpointing does incur an overhead. For the case of $p = 2$ remote computers, we provide schedules whose expected work completion is asymptotically optimal, as the size of the workload grows without bound; we also provide some guidelines for deriving exactly optimal schedules (Section 4). The complexity of the development in Section 4 suggests that the general case of p remote computers will be prohibitively difficult, even with respect to deriving asymptotically optimal schedules. Therefore, we settle in this general case for deriving a number of well-structured heuristics, whose quality can be assessed via explicit expressions for their expected work outputs (Section 5). Simulations suggest that one of our six heuristics—regrettably, the computationally most complicated one—is the clear winner in terms of performance. An extensive suite of simulation experiments suggests that all of our heuristics provide schedules with good expected work output, and that the "clear winner" in the competition of Section 5 does, indeed, dominate the others (Section 6). We extended to general risk functions the asymptotic optimality result for two computers and then the p-computer heuristics. Extensive simulations using actual traces of computers' availabilities suggest that the clear winner of Sections 5 and 6 also dominate other solutions in the presence of general risk functions (Section 7).

Much remains to be done regarding this important problem, but three directions stand out as perhaps the major outstanding challenges. One of these is to extend our study to include heterogeneous assemblages of remote computers, whose constituent computers differ in speed and other computational resources. When the assemblages are heterogeneous, but even when they are homogeneous, it would be significant to allow the assemblage's computers to be subject to differing probabilities of being interrupted.

Acknowledgment. The work of A. Benoit and Y. Robert was supported in part by the ANR StochaGrid project. The work of A. Rosenberg was supported in part by US NSF Grants CNS-0842578 and CNS-0905399.

The authors thank Joshua Wingstrom who give them access to the availability traces.

References

[1] Micah Adler, Ying Gong, and Arnold L. Rosenberg. Asymptotically optimal worksharing in HNOWs: How long is "sufficiently long?". In *36th Annual Simulation Symposium*, pages 39–46, 2003.

[2] Micah Adler, Ying Gong, and Arnold L. Rosenberg. On "exploiting" node-heterogeneous clusters optimally. *Theory of Computing Systems*, 42(4):465–487, 2008.

[3] Cosimo Anglano, John Brevik, Massimo Canonico, Dan Nurmi, and Rich Wolski. Fault-aware scheduling for bag-of-tasks applications on desktop grids. In *GRID '06*, pages 56–63. IEEE Computer Society, 2006.

[4] Remzi H. Arpaci, Andrea C. Dusseau, Amin M. Vahdat, Lok T. Liu, Thomas E. Anderson, and David A. Patterson. The interaction of parallel and sequential workloads on a network of workstations. In *SIGMETRICS '95/PERFORMANCE '95*, pages 267–278. ACM, 1995.

[5] Baruch Awerbuch, Yossi Azar, Amos Fiat, and Frank Thomson Leighton. Making commitments in the face of uncertainty: How to pick a winner almost every time. In *28th ACM SToC*, pages 519–530, 1996.

[6] Mohammad Banikazemi, Vijay Moorthy, and Dhabaleswar K. Panda. Efficient collective communication on heterogeneous networks of workstations. In *Intl. Conf. on Parallel Processing (ICPP)*, pages 460–467, 1998.

[7] Cyril Banino, Olivier Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, and Yves Robert. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. *IEEE Trans. Parallel Distributed Systems*, 15(4):319–330, 2004.

[8] Gerassimos D. Barlas. Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of a divisible load on arbitrary processor trees. *IEEE Trans. Parallel Distrib. Syst.*, 9(5):429–441, 1998.

[9] Olivier Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, and Yves Robert. Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In *16th Intl. Parallel and Distributed Processing Symp. (IPDPS)*. IEEE Computer Society Press, 2002.

[10] Olivier Beaumont, Loris Marchal, and Yves Robert. Scheduling divisible loads with return messages on heterogeneous master-worker platforms. In *Intl. Conf. on High Performance Computing HiPC'2005*, volume 3769 of *LNCS*, pages 498–507. Springer Verlag, 2005.

[11] Michael A. Bender and Cynthia A. Phillips. Scheduling dags on asynchronous processors. In *19th ACM SPAA*, pages 35–45, 2007.

[12] Veeravalli Bharadwaj, Debasish Ghose, Venkataraman Mani, and Thomas G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. Wiley-IEEE Computer Society Press, 1996.

[13] Sandeep N. Bhatt, Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. An optimal strategies for cycle-stealing in networks of workstations. *IEEE Trans. Computers*, 46(5):545–557, 1997.

[14] Rajkumar Buyya, David Abramson, and Jonathan Giddy. A case for economy grid architecture for service-oriented grid computing. In *10th Heterogeneous Computing Workshop*. IEEE Computer Society, 2001.

[15] Y.-C. Cheng and T.G. Robertazzi. Distributed computation for a tree network with communication delays. *IEEE Transactions on Aerospace and Electronic Systems*, 26(3):511–516, 1990.

[16] Walfredo Cirne and Keith Marzullo. The computational co-op: Gathering clusters into a metacomputer. In *13th Intl. Parallel Processing Symp. (IPPS)*, pages 160–166, 1999.

[17] Pierre-Franɛcois Dutot. Master-slave tasking on heterogeneous processors. In *17th Intl. Parallel and Distributed Processing Symp. (IPDPS)*, 2003.

[18] Ian Foster and Carl Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 2004.

[19] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Intl. J. High Performance Computing Applications*, 15(3):200–222, 2001.

[20] Pierre Fraigniaud, Bernard Mans, and Arnold L. Rosenberg. Efficient trigger-broadcasting in heterogeneous clusters. *J. Parallel Distrib. Comput.*, 65(5):628–642, 2005.

[21] Li Gao and Grzegorz Malewicz. Toward maximizing the quality of results of dependent tasks computed unreliably. *Theory Comput. Syst.*, 41(4):731–752, 2007.

[22] Ram Kesavan, Kiran Bondalapati, and Dhabaleswar K. Panda. Multicast on irregular switch-based networks with wormhole routing. In *3rd IEEE Symp. on High-Performance Computer Architecture*, pages 48–57, 1997.

[23] Derrick Kondo. *Scheduling Task Parallel Applications For Rapid Turnaround on Enterprise Desktop Grids*. PhD thesis, University of California at San Diego, July 2005.

[24] Derrick Kondo, Henri Casanova, Eric Wing, and Francine Berman. Models and scheduling mechanisms for global computing applications. In *16th Intl. Parallel and Distr. Processing Symp. (IPDPS)*, 2002.

[25] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky. Seti@home-massively distributed computing for seti. *Computing in Science & Engineering*, 3(1):78–83, 2001.

[26] Michael J. Litzkow, Miron Livny, and Matt W. Mutka. Condor - a hunter of idle workstations. In *ICDCS*, pages 104–111, 1988.

[27] Grzegorz Malewicz, Arnold L. Rosenberg, and Matthew Yurkewych. Toward a theory for scheduling dags in internet-based computing. *IEEE Trans. Computers*, 55(6):757–768, 2006.

[28] Daniel Nurmi, John Brevik, and Rich Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-Par 2005 Parallel Processing*, volume 3648 of *LNCS*, pages 432–441, 2005.

[29] Gregory F. Pfister. *In Search of Clusters*. Prentice-Hall, 1995.

[30] J.S. Plank and W.R. Elwasif. Experimental assessment of workstation failures and their impact on checkpointing systems. In *Fault-Tolerant Computing, 1998*, pages 48–57, June 1998.

[31] Arnold L. Rosenberg. Guidelines for data-parallel cycle-stealing in networks of workstations i: On maximizing expected output. *J. Parallel Distrib. Comput.*, 59(1):31–53, 1999.

[32] Arnold L. Rosenberg. Guidelines for data-parallel cycle-stealing in networks of workstations ii: On maximizing guaranteed output. *Intl. J. Foundations of Computer Science*, 11(1):183–204, 2000.

[33] Arnold L. Rosenberg. Sharing partitionable workloads in heterogeneous nows: Greedier is not better. In *IEEE Intl. Conf. on Cluster Computing (CLUSTER 2001)*, pages 124–131, 2001.

[34] Arnold L. Rosenberg. Optimal schedules for cycle-stealing in a network of workstations with a bag-of-tasks workload. *IEEE Trans. Parallel Distrib. Syst.*, 13(2):179–191, 2002.

[35] Arnold L. Rosenberg. Changing challenges for collaborative algorithmics. In A. Zomaya, editor, *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*, pages 1–44. Springer, 2006.

[36] Bharadwaj Veeravalli, Debasish Ghose, and V. Mani. Optimal sequencing and arrangement in distributed single-level tree networks with communication delays. *IEEE Trans. Parallel Distrib. Syst.*, 5(9):968–976, 1994.

[37] Bharadwaj Veeravalli, Debasish Ghose, and V. Mani. Multi-installment load distribution in tree networks with delays. *IEEE Transactions on Aerospace and Electronic Systems*, 31(2):555–567, 1995.

[38] S.W. White and D.C. Torney. Use of a workstation cluster for the physical mapping of chromosomes. *SIAM NEWS*, pages 14–17, March 1993.

[39] Joshua Wingstrom and Henri Casanova. Probabilistic allocation of tasks on desktop grids. In *Proceedings of PCGrid*. IEEE CS Press, 2008.

[40] Yang Yang and Henri Casanova. Umr: A multi-round algorithm for scheduling divisible workloads. In *17th Intl. Parallel and Distributed Processing Symp. (IPDPS)*, page 24, 2003.

A Experiments with linear risk functions (selected heuristics)

On the following graphs, the only group-heuristic whose performance is reported is Σ_{greedy}.

A.1 Experiments E1

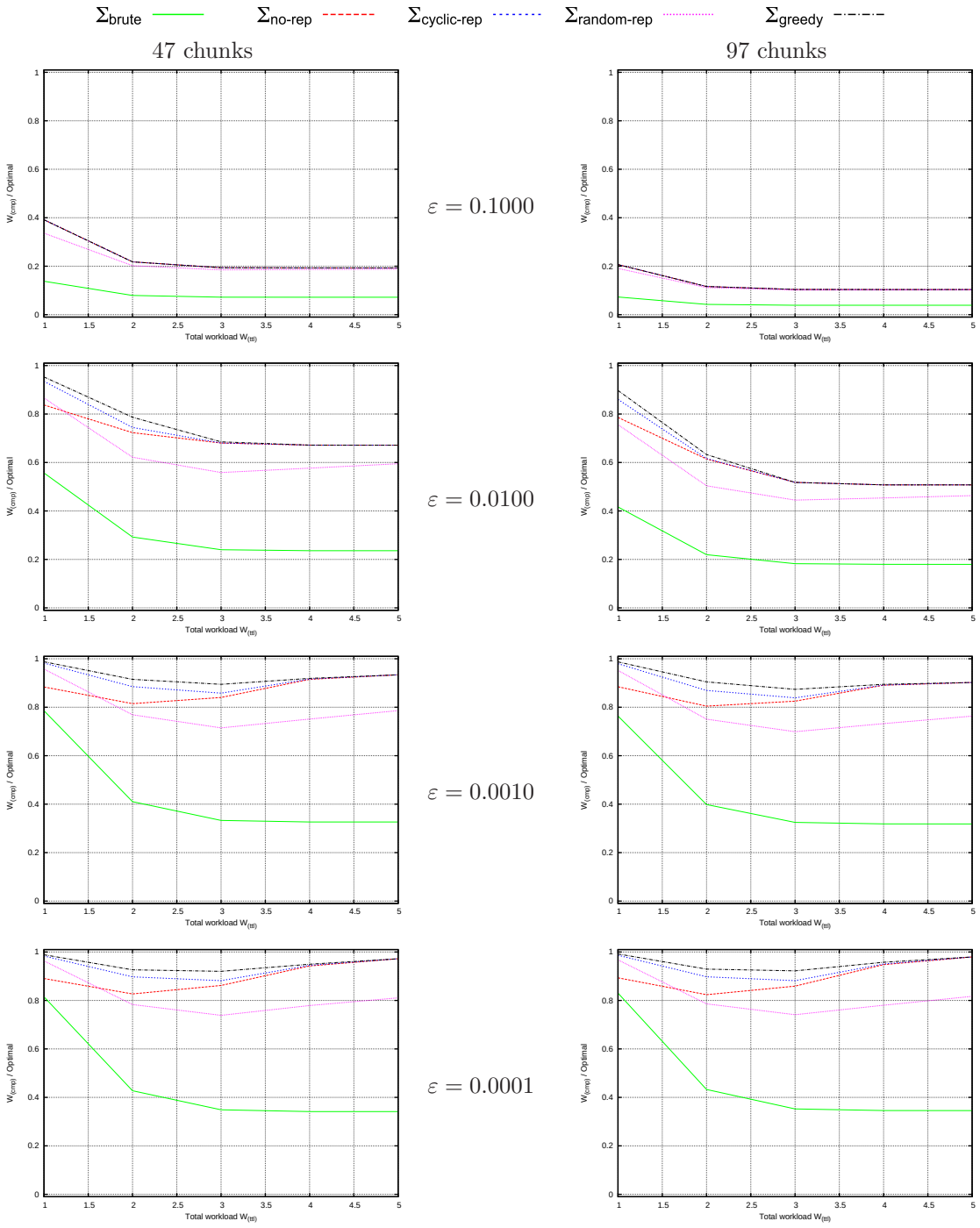


Figure 14: Experiment (E1) using 5 computers.

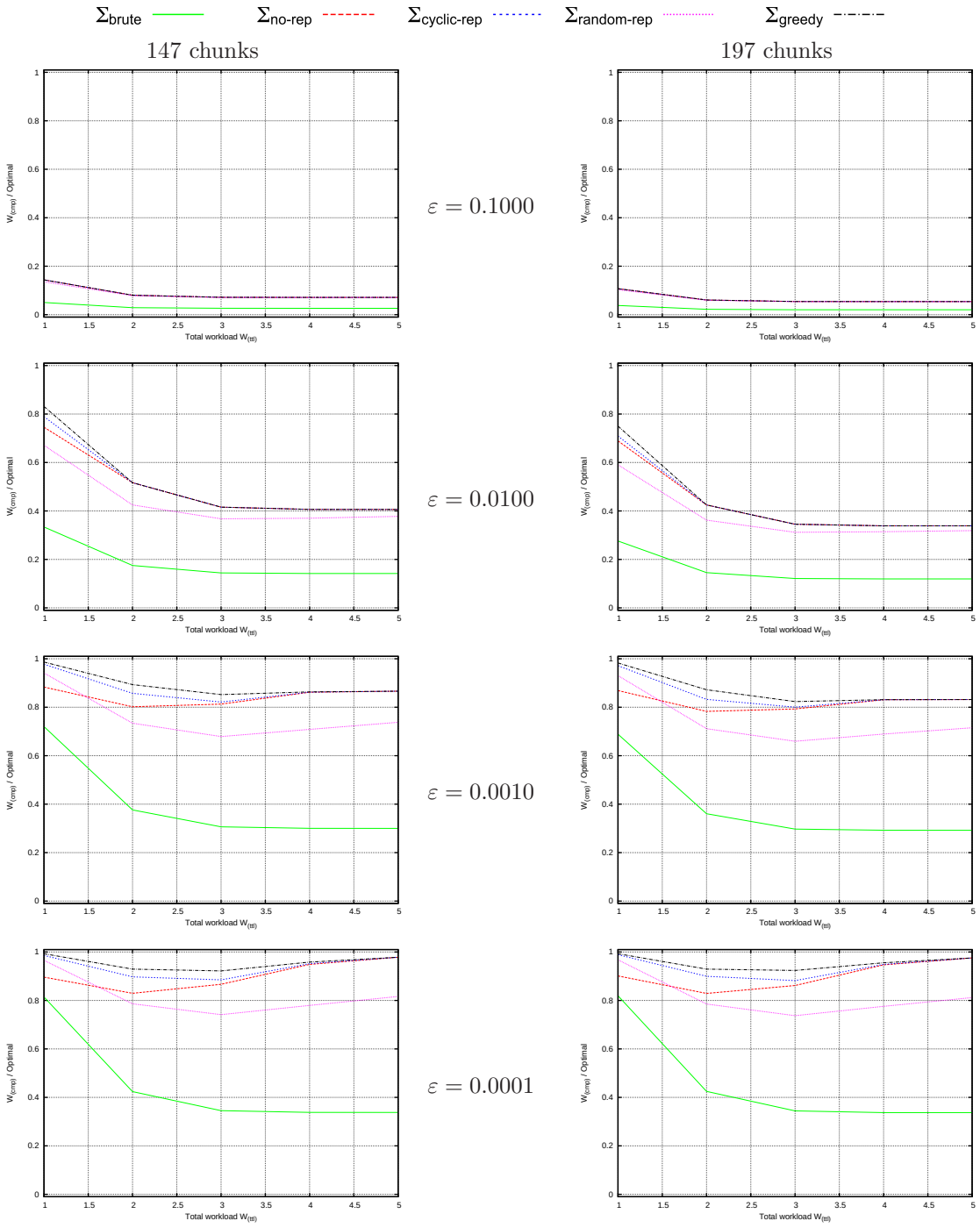


Figure 15: Experiment (E1) using 5 computers (continued).

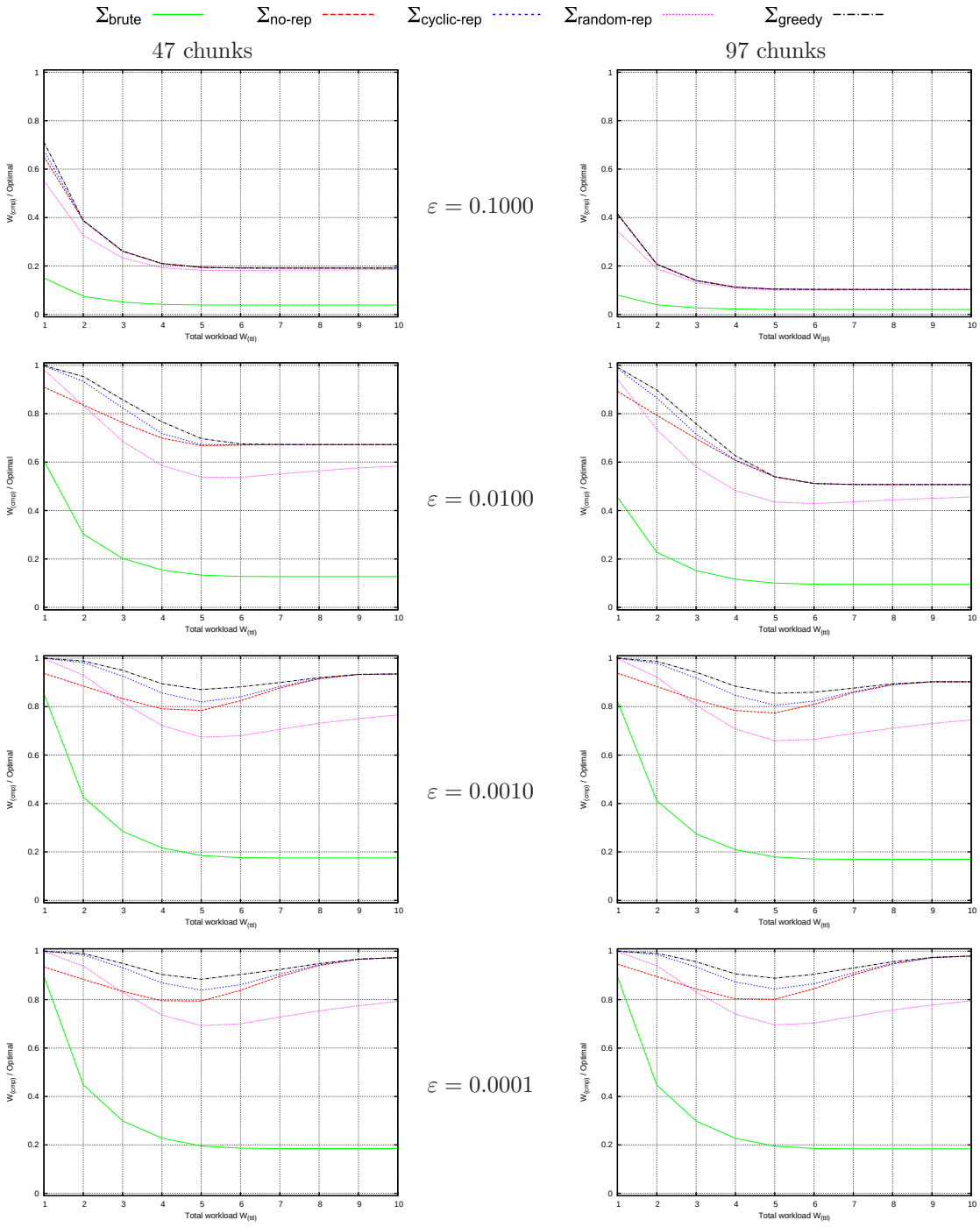


Figure 16: Experiment (E1) using 10 computers.

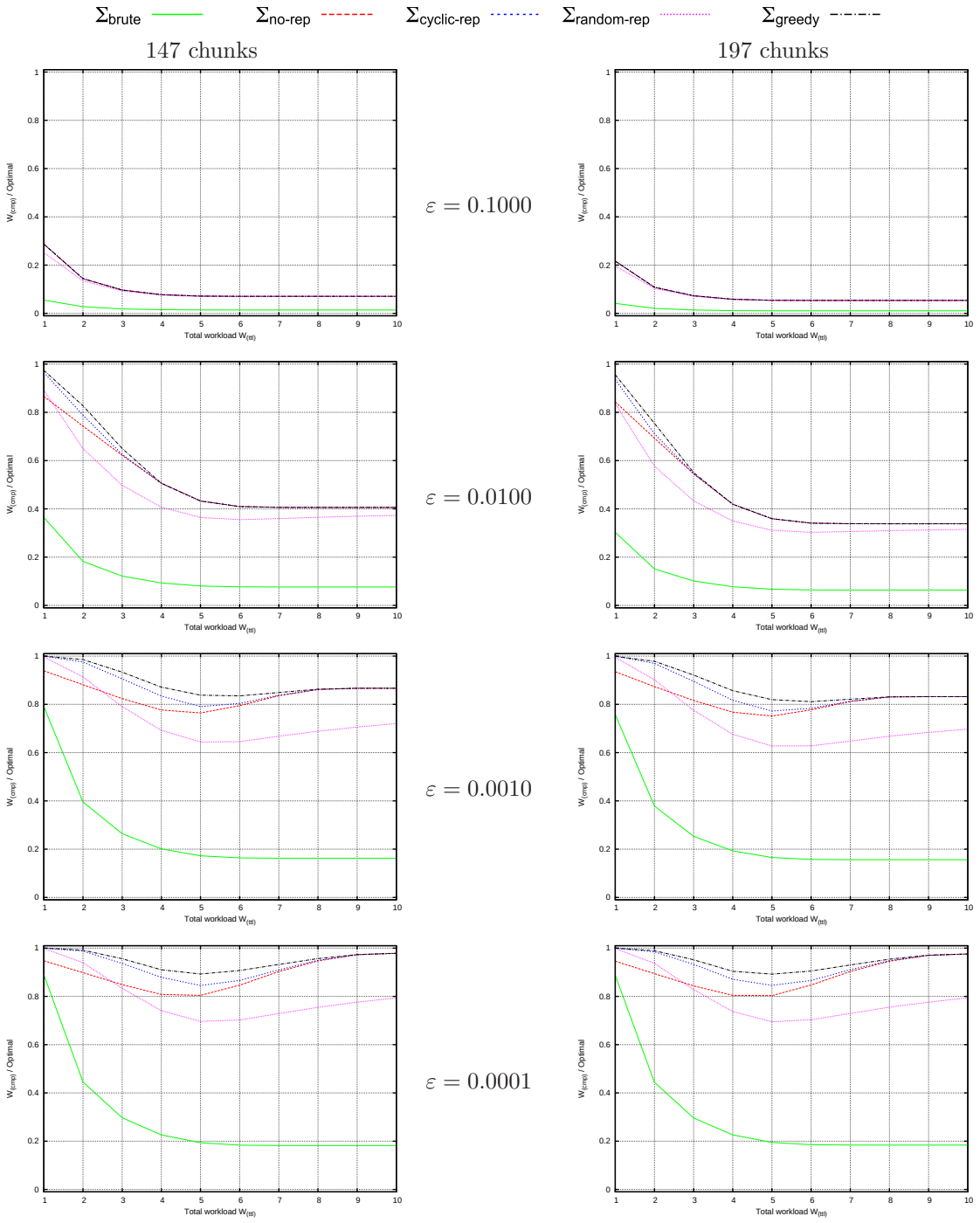


Figure 17: Experiment (E1) using 10 computers (continued).

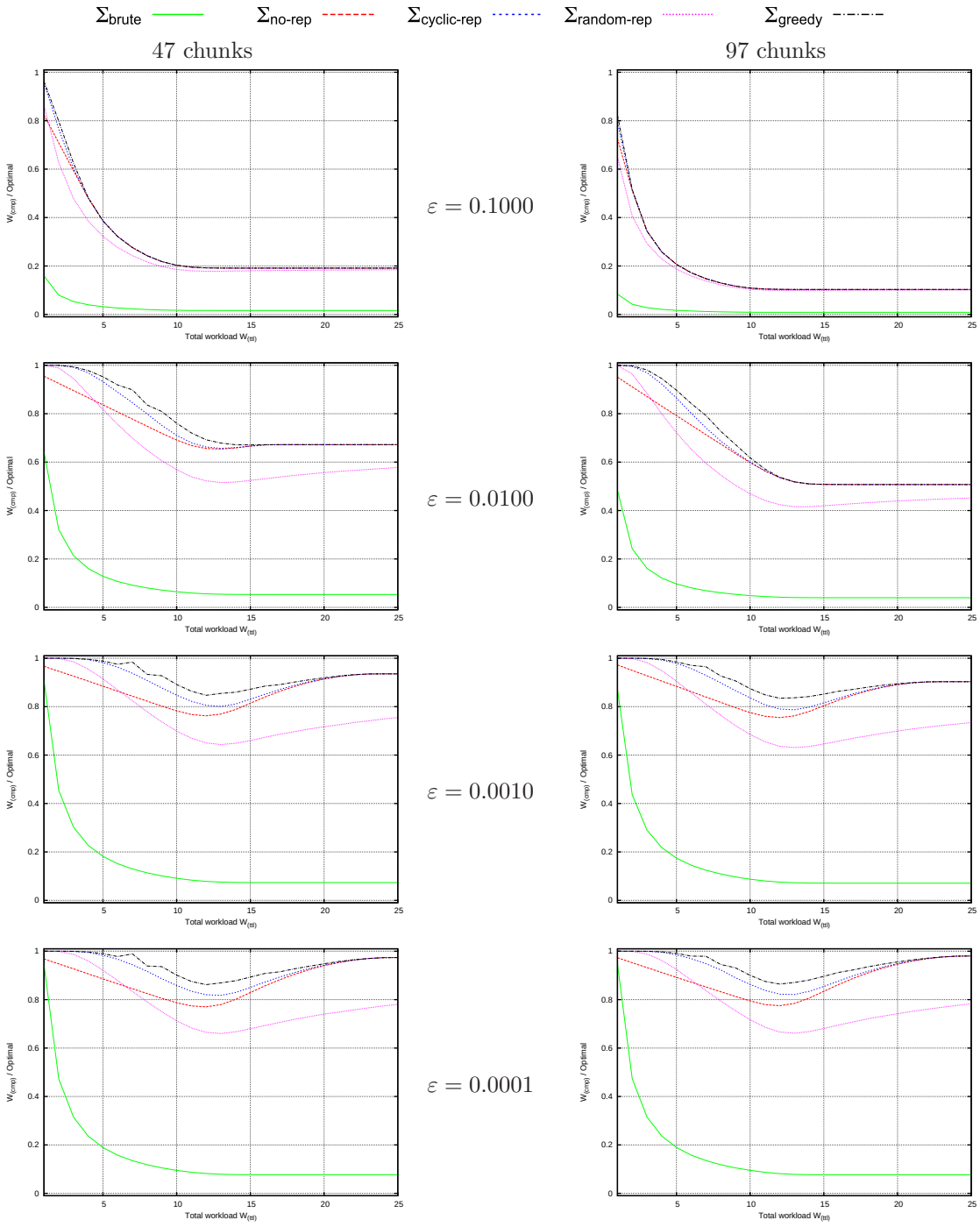


Figure 18: Experiment (E1) using 25 computers.

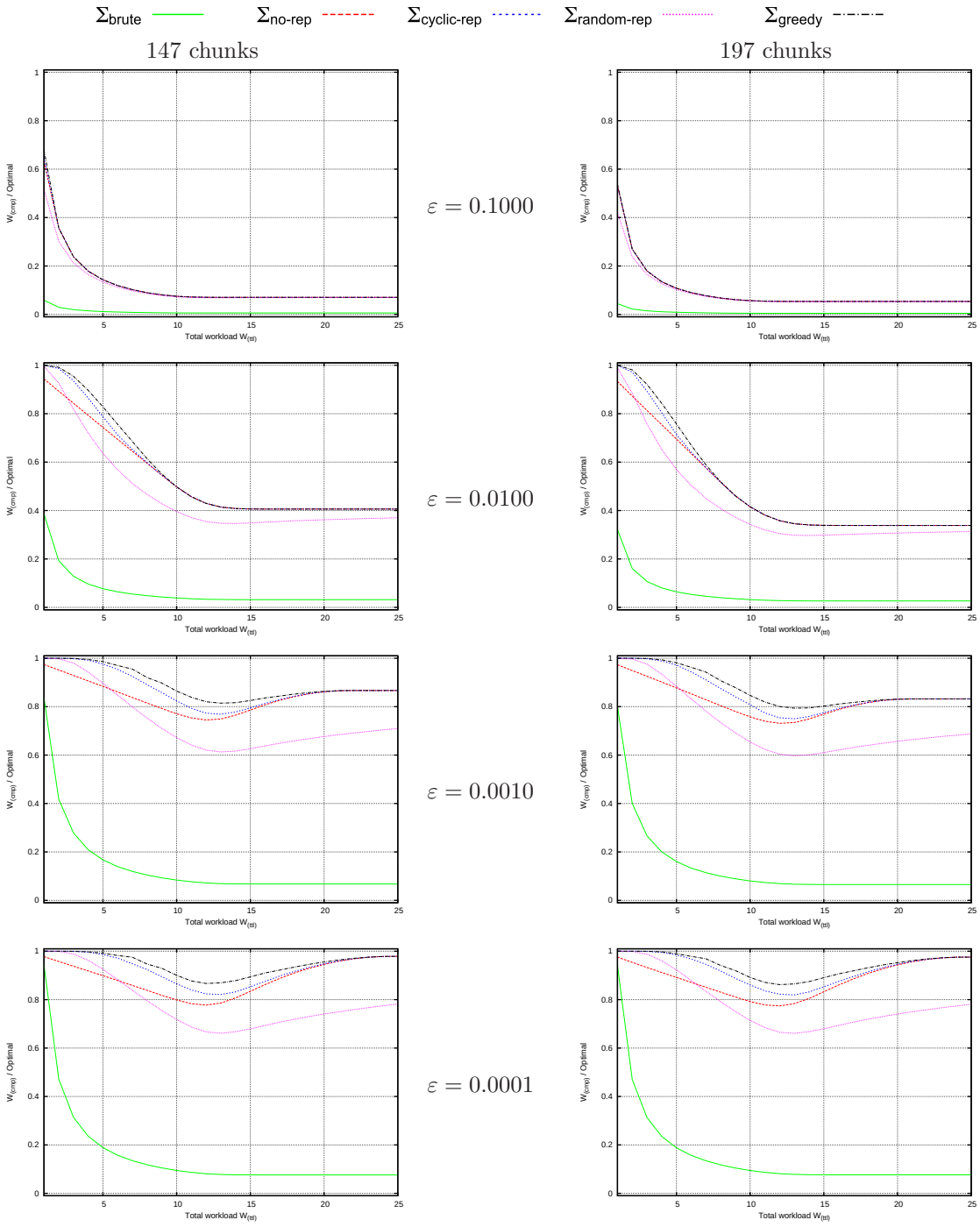


Figure 19: Experiment (E1) using 25 computers (continued).

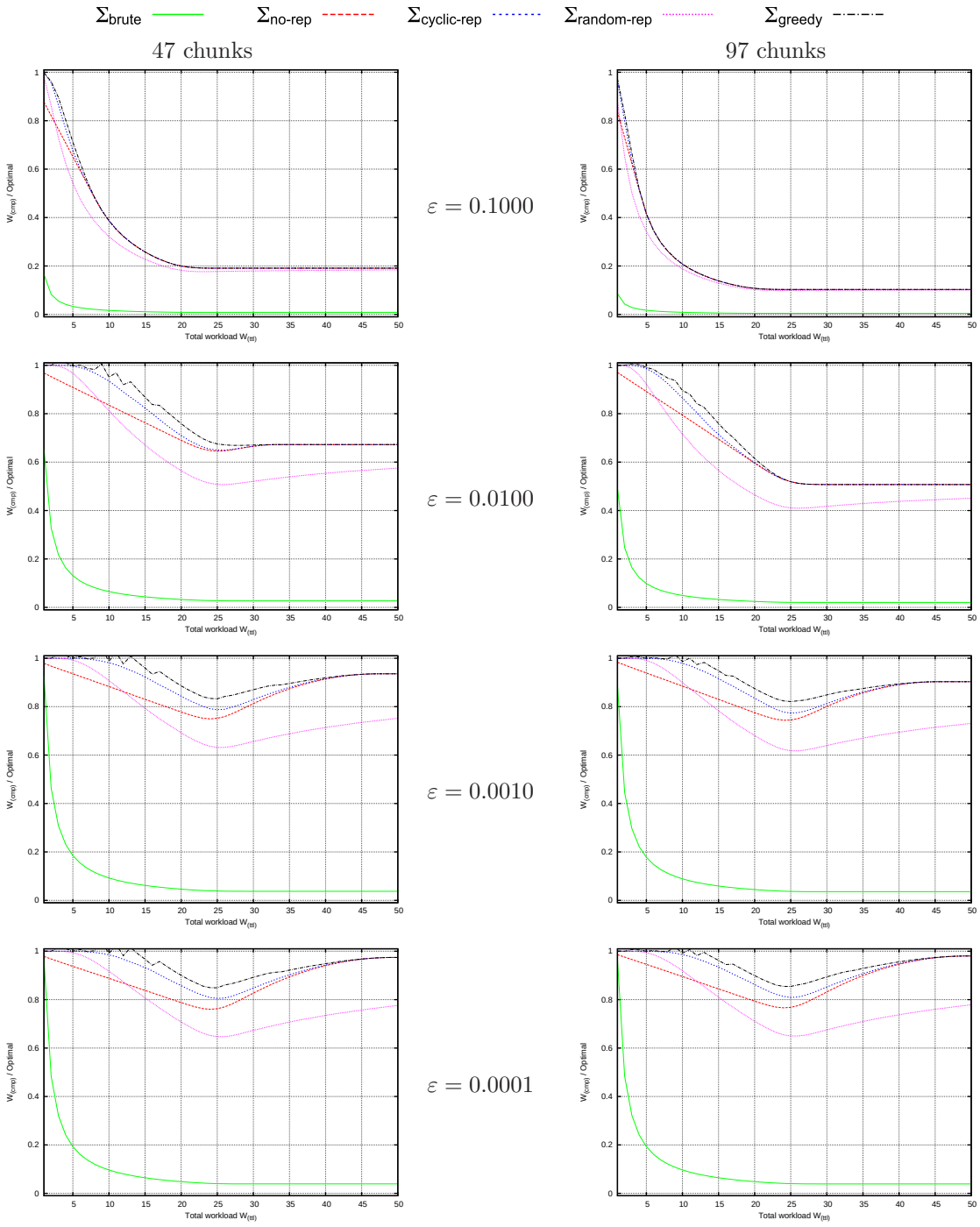


Figure 20: Experiment (E1) using 50 computers.

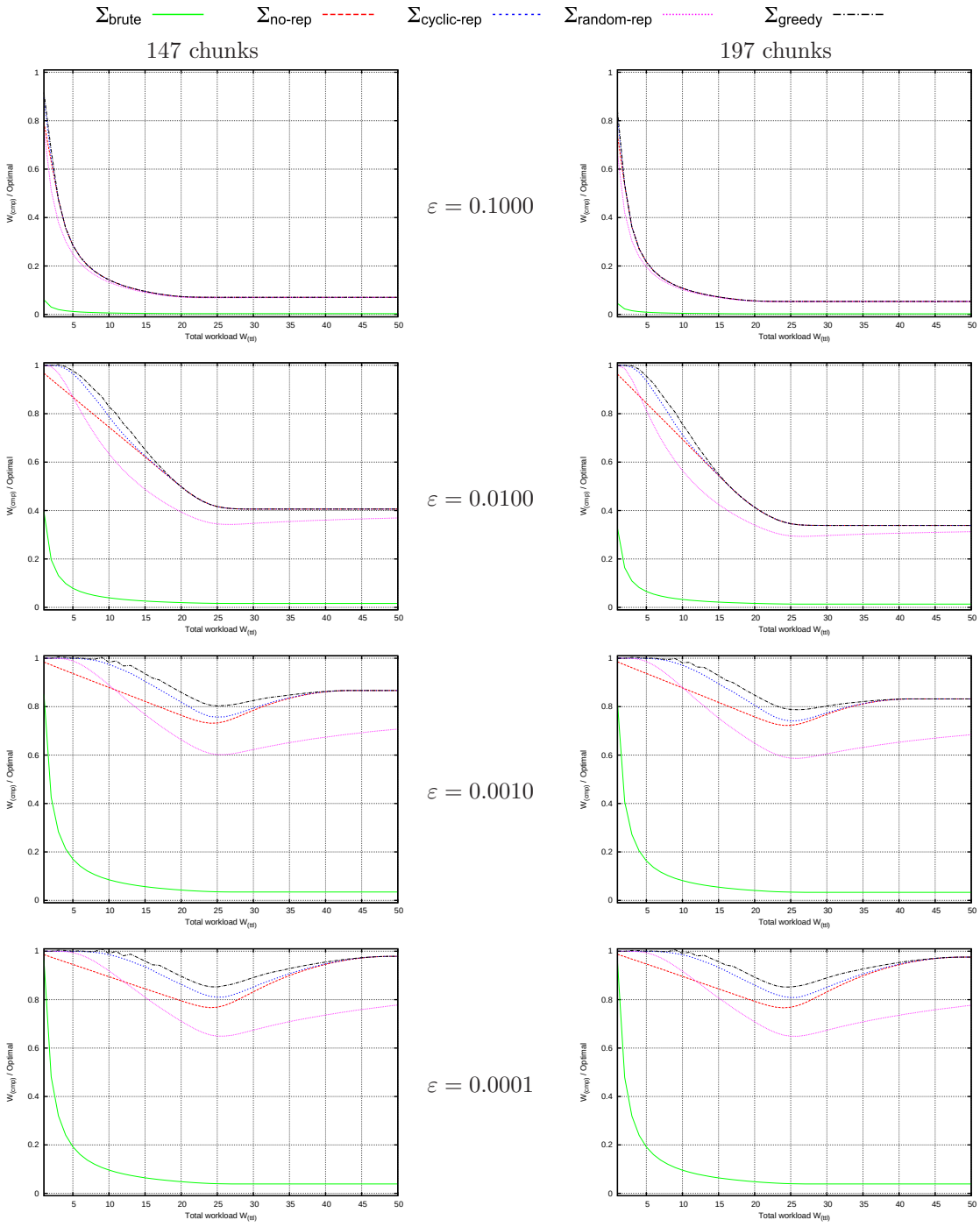


Figure 21: Experiment (E1) using 50 computers (continued).

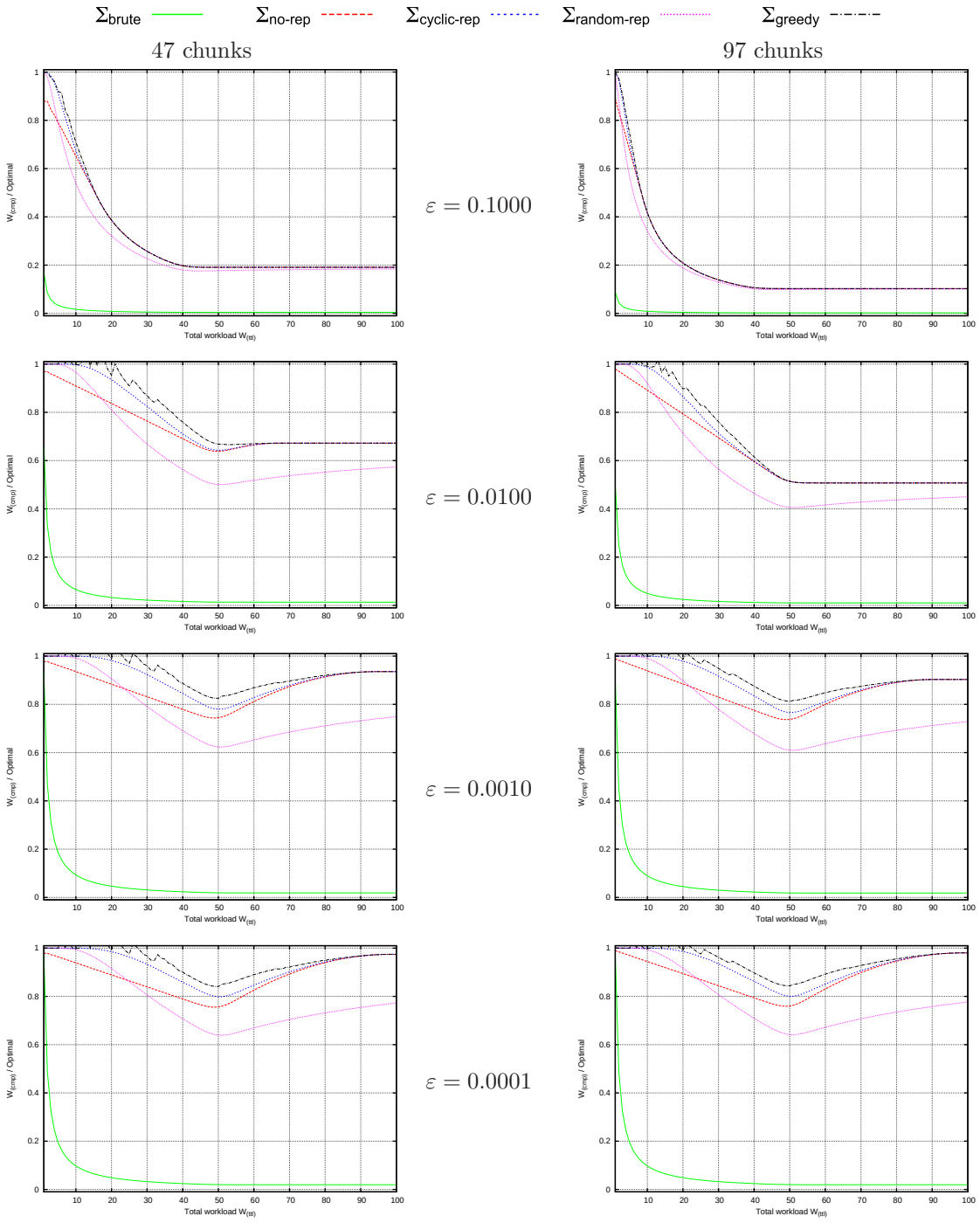


Figure 22: Experiment (E1) using 100 computers.

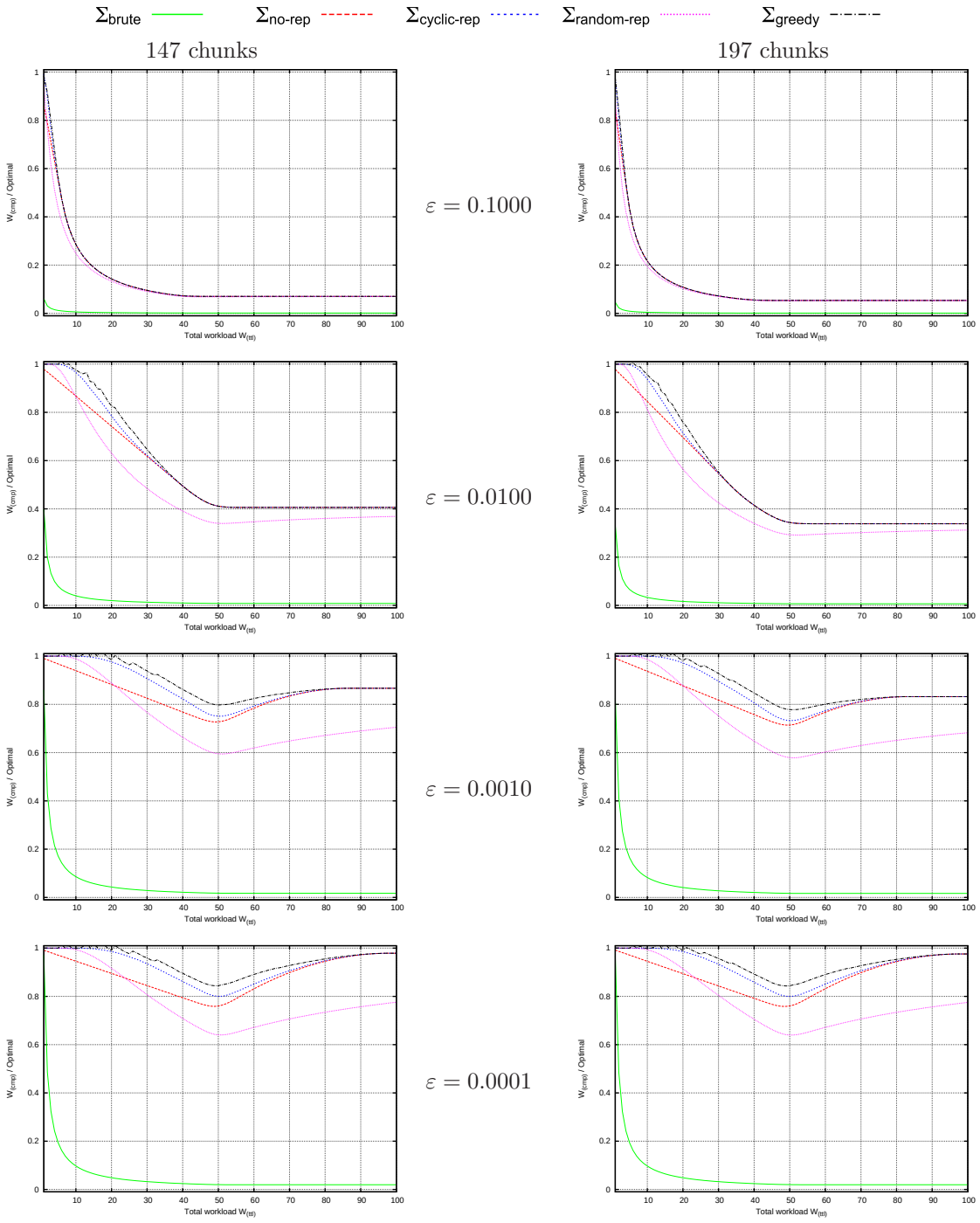


Figure 23: Experiment (E1) using 100 computers (continued).

A.2 Experiments E2

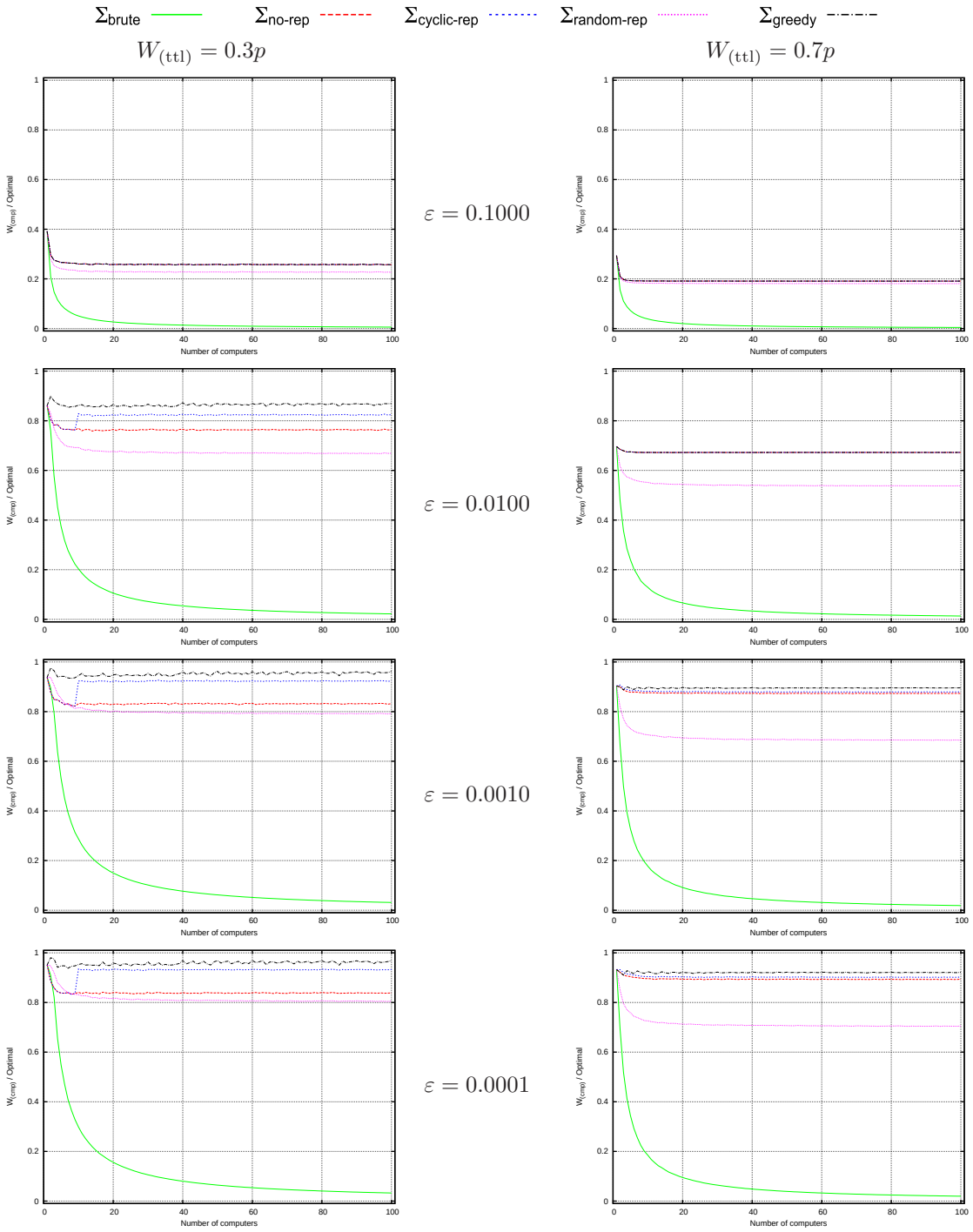


Figure 24: Experiment (E2) with 47 chunks.

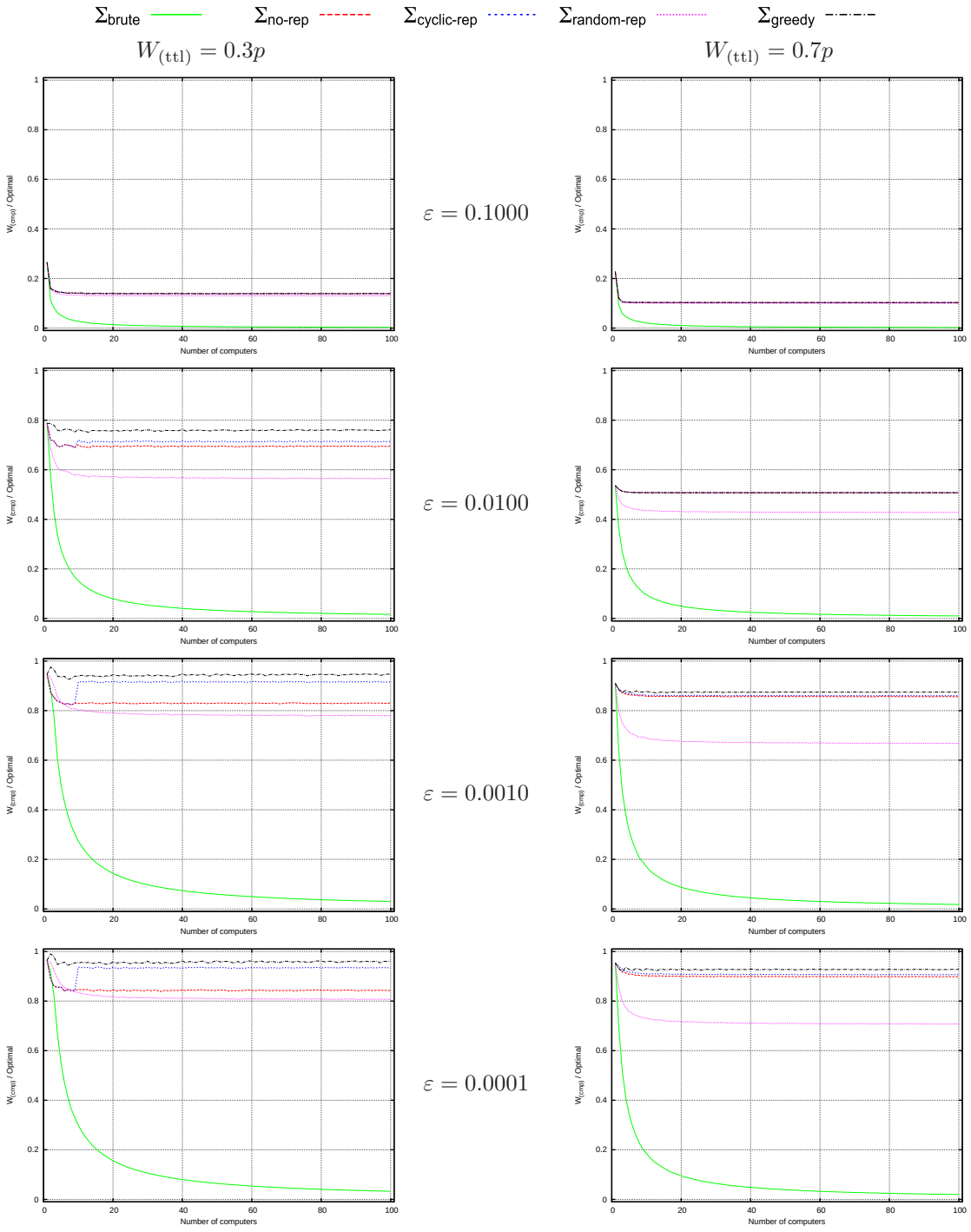


Figure 25: Experiment (E2) with 97 chunks.

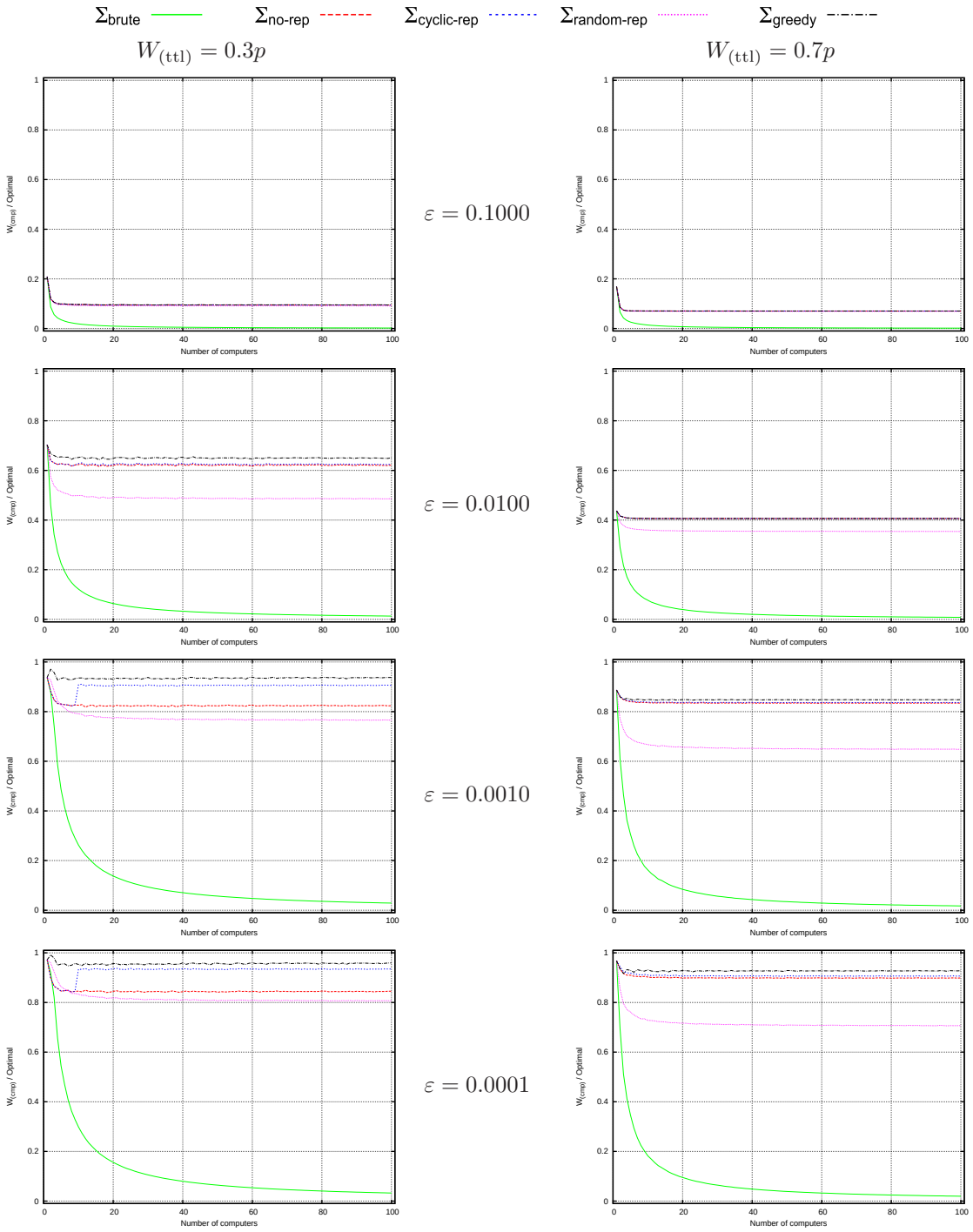


Figure 26: Experiment (E2) with 147 chunks.

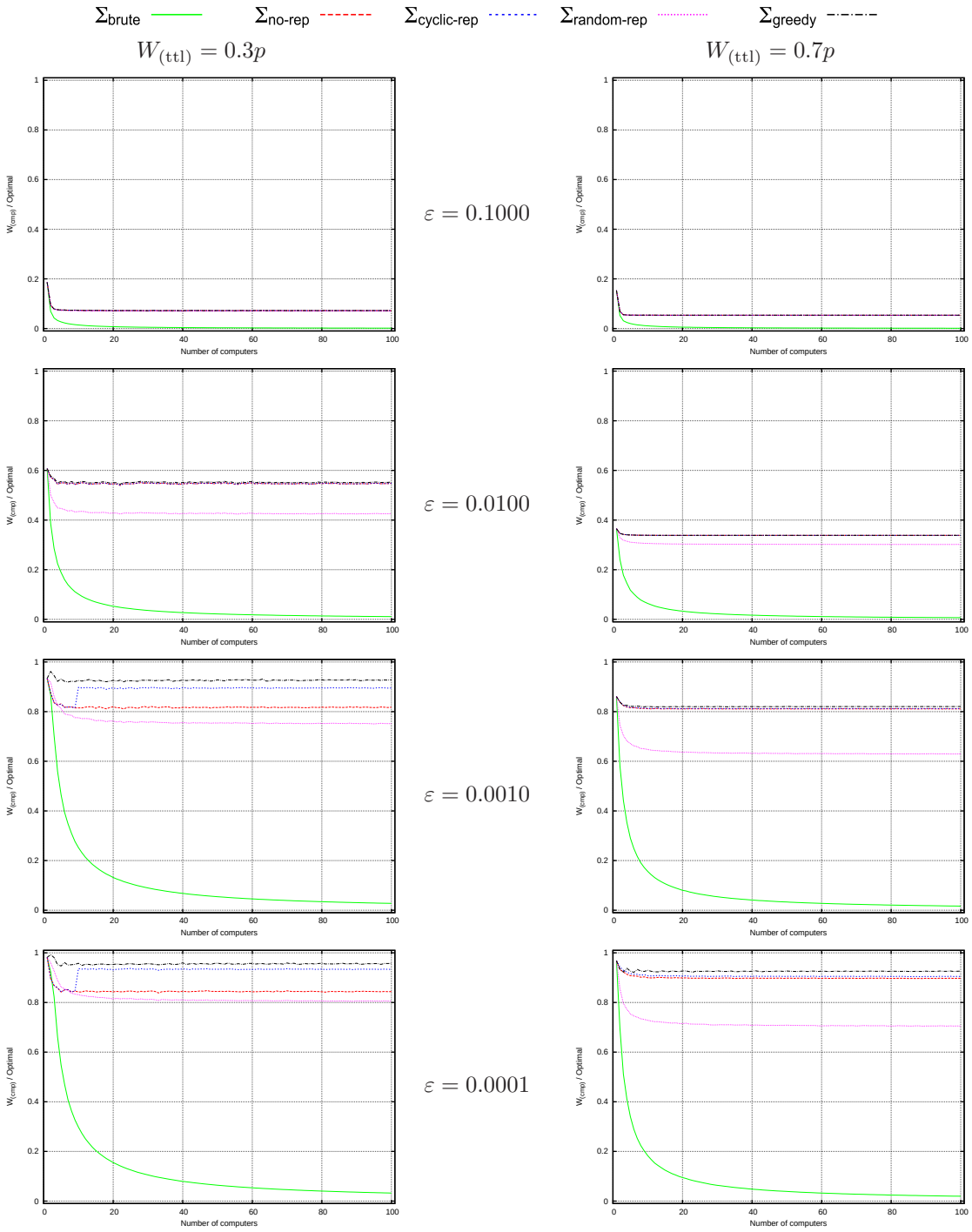


Figure 27: Experiment (E2) with 197 chunks.

A.3 Experiments E3

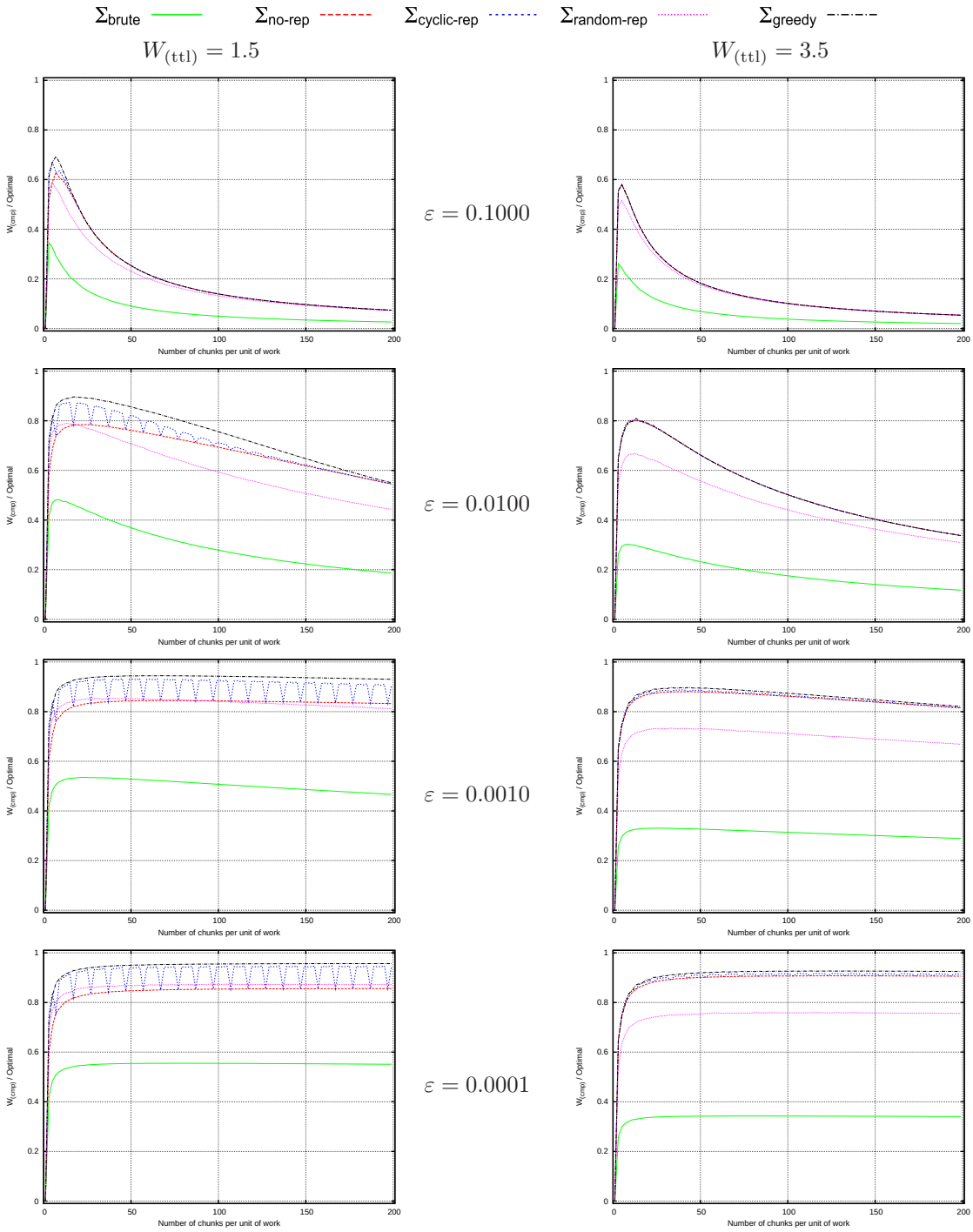


Figure 28: Experiment (E3) using 5 computers.

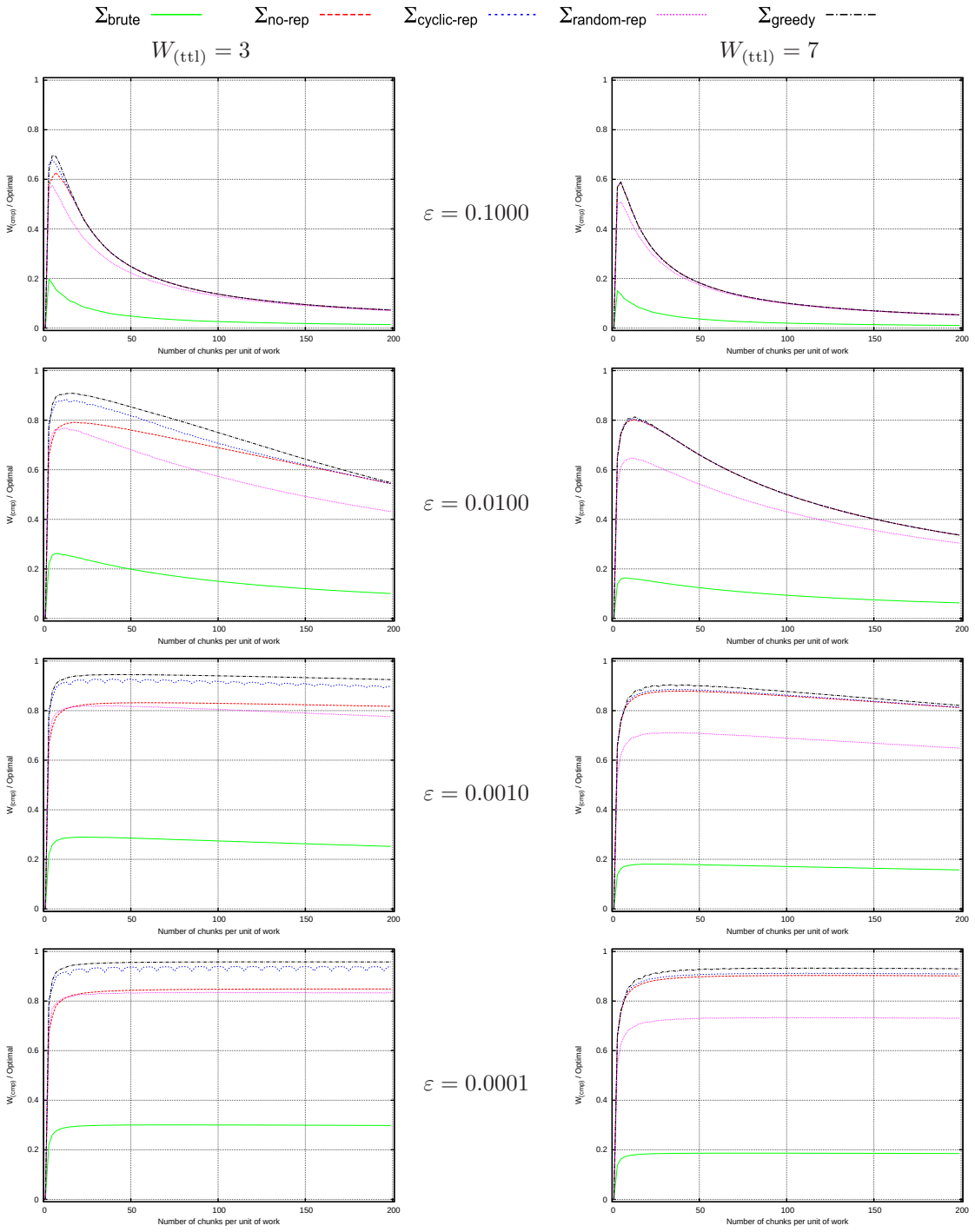


Figure 29: Experiment (E3) using 10 computers.

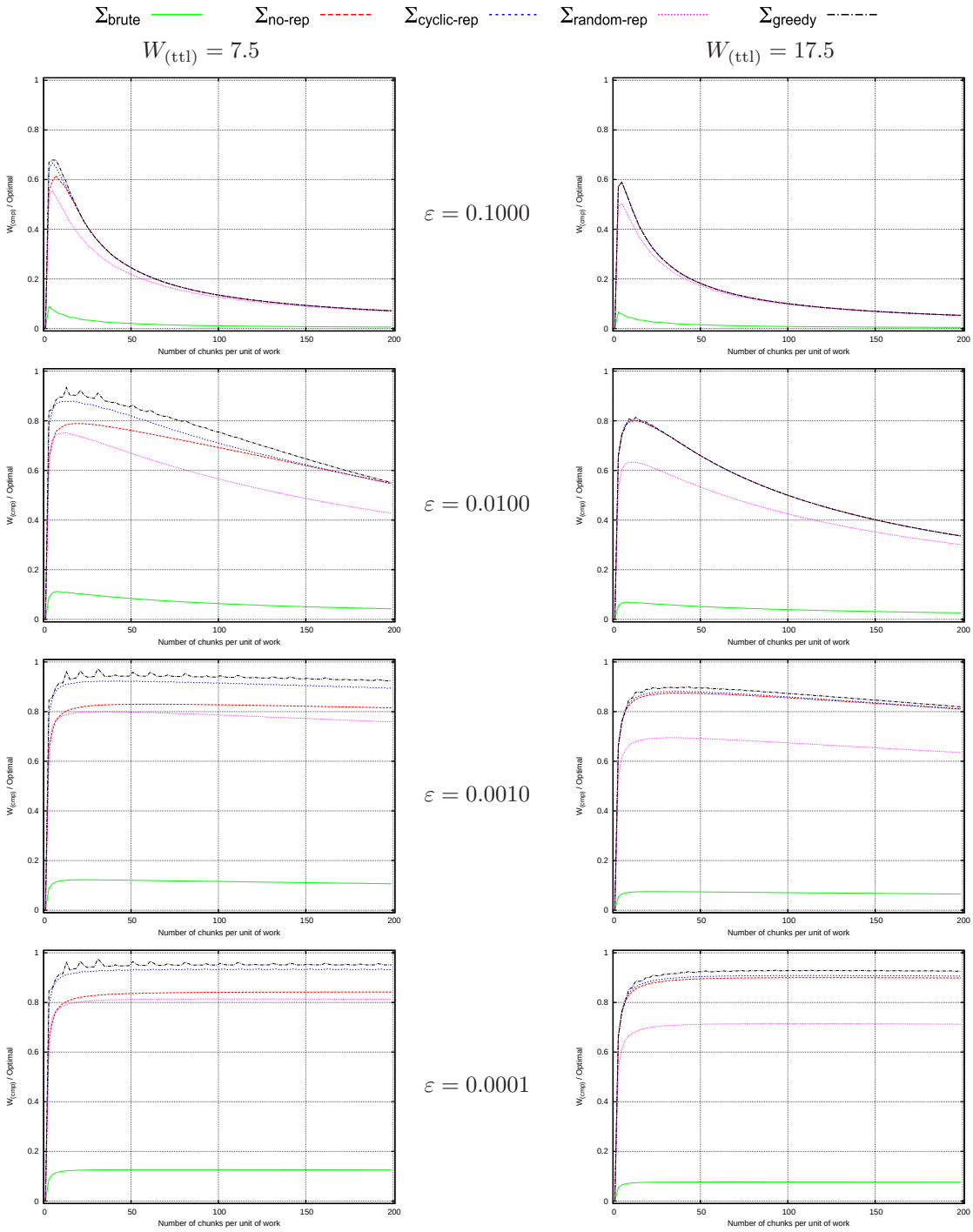


Figure 30: Experiment (E3) using 25 computers.

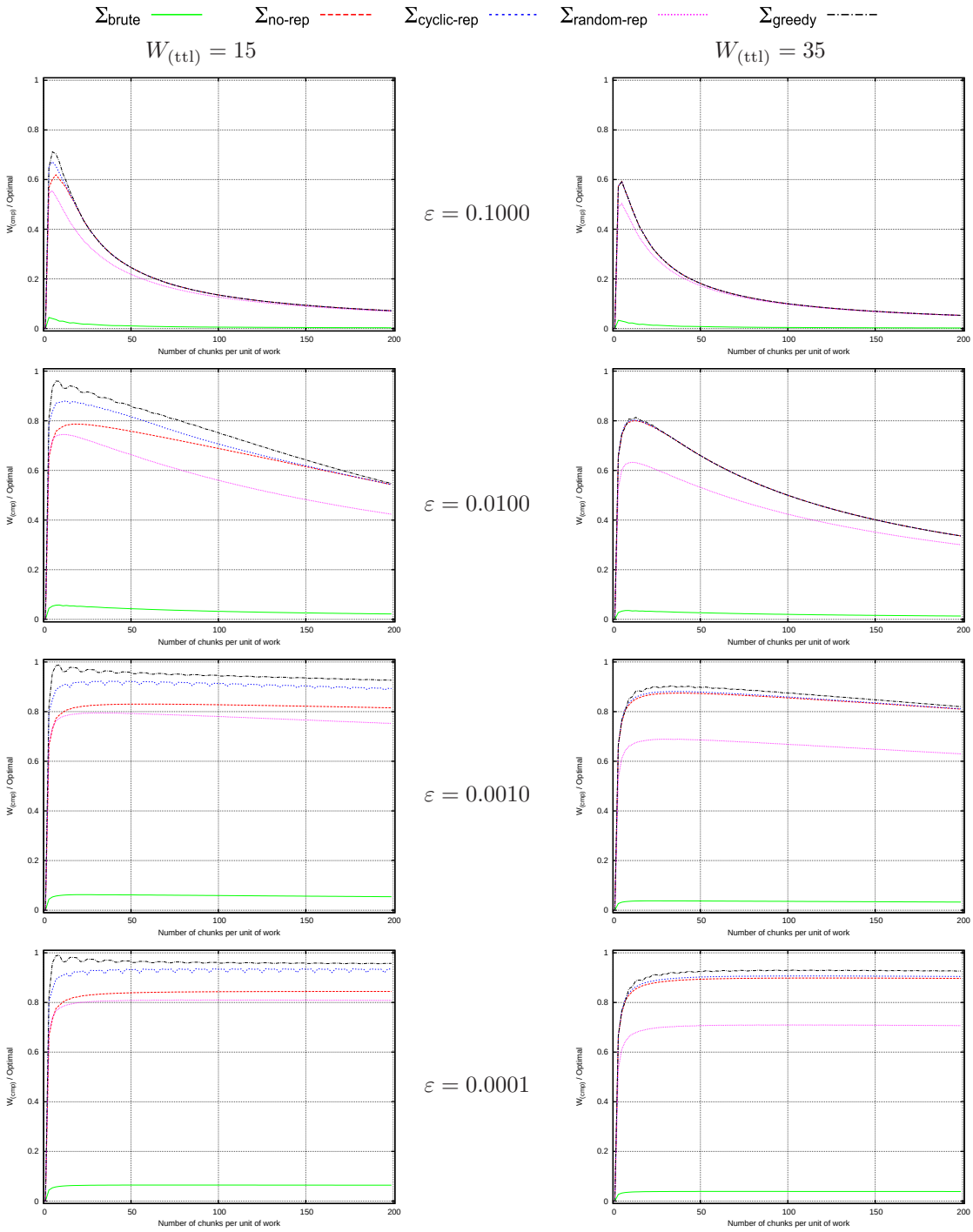


Figure 31: Experiment (E3) using 50 computers.

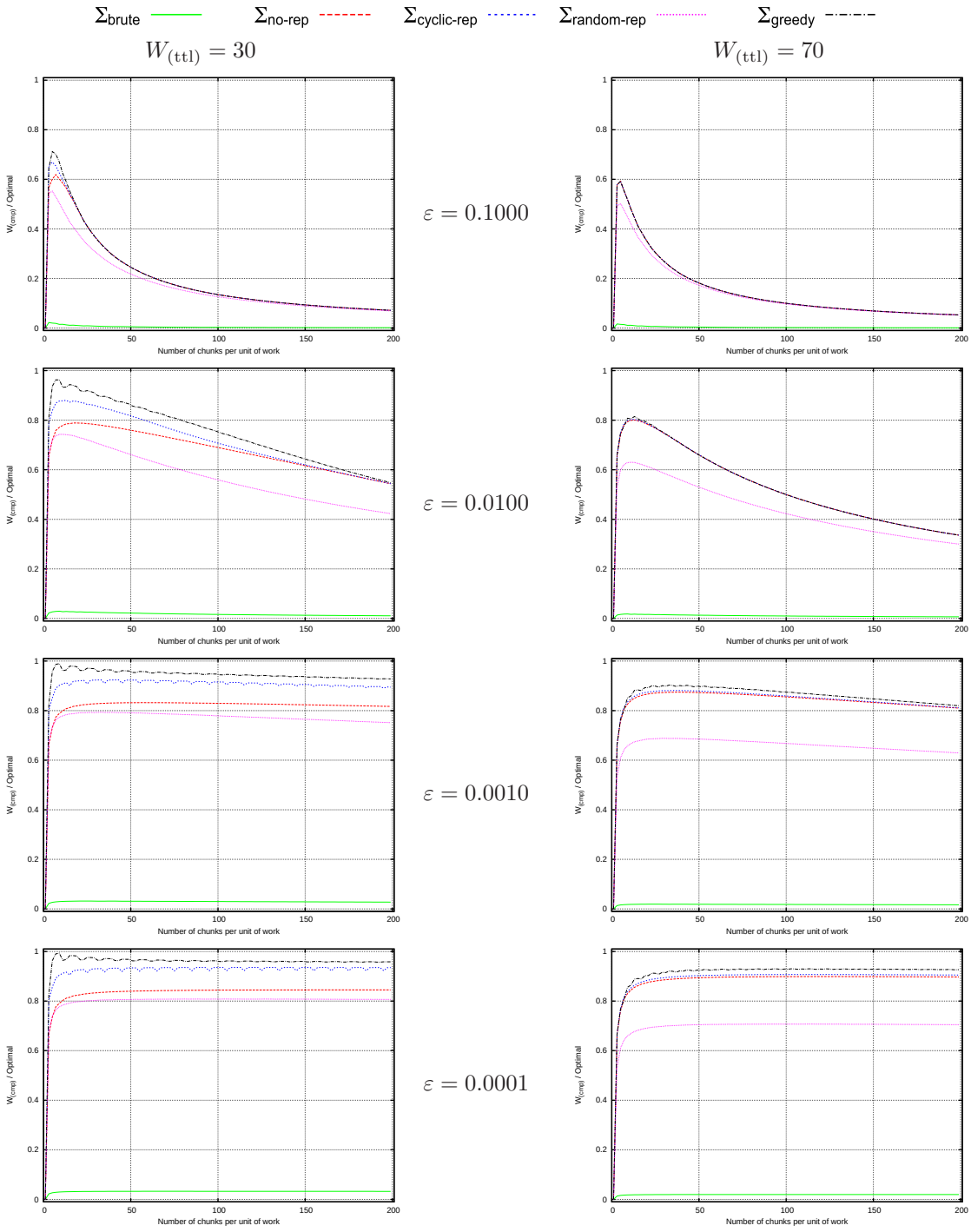


Figure 32: Experiment (E3) using 100 computers.

A.4 Experiments E4

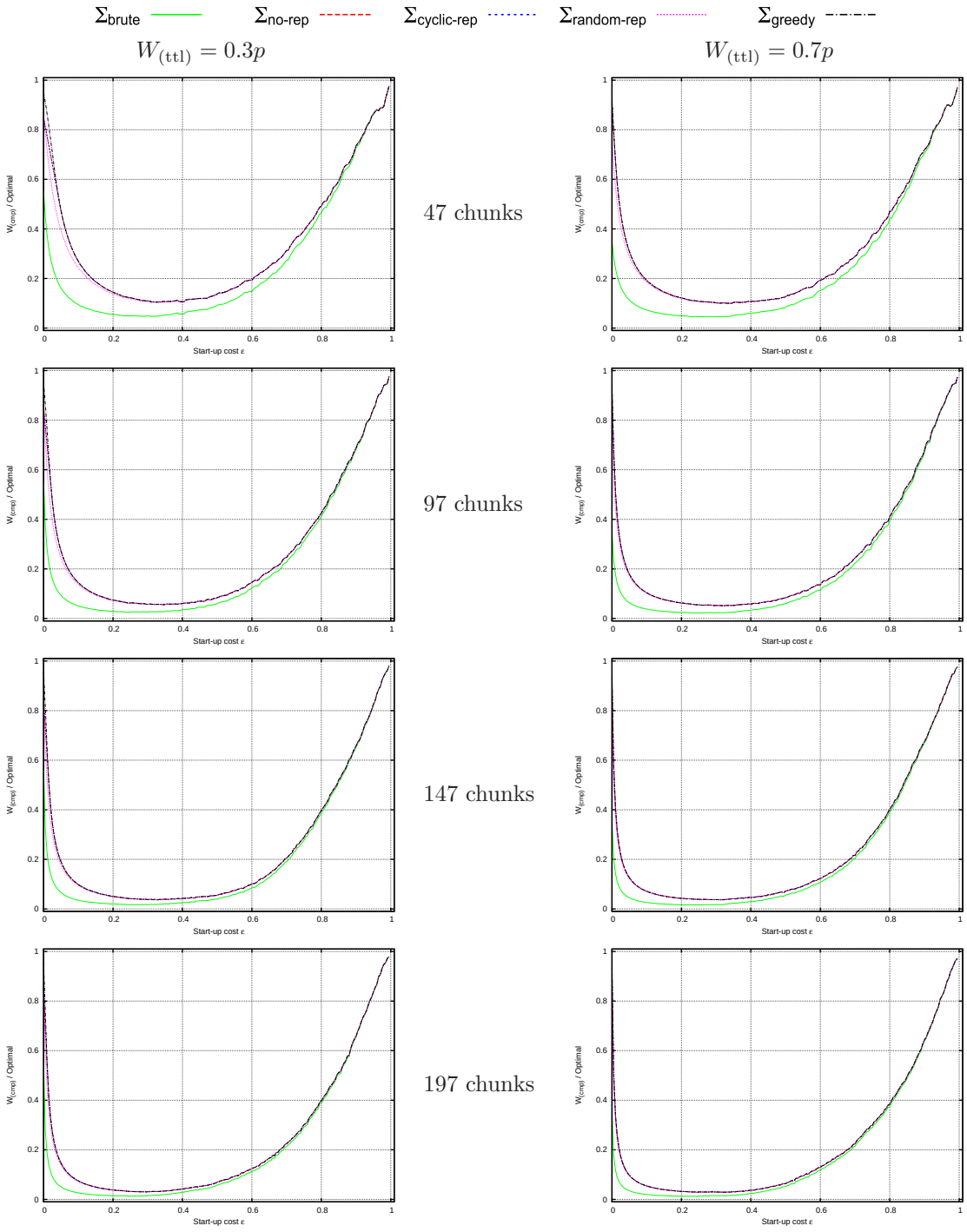


Figure 33: Experiment (E4) with 5 computers.

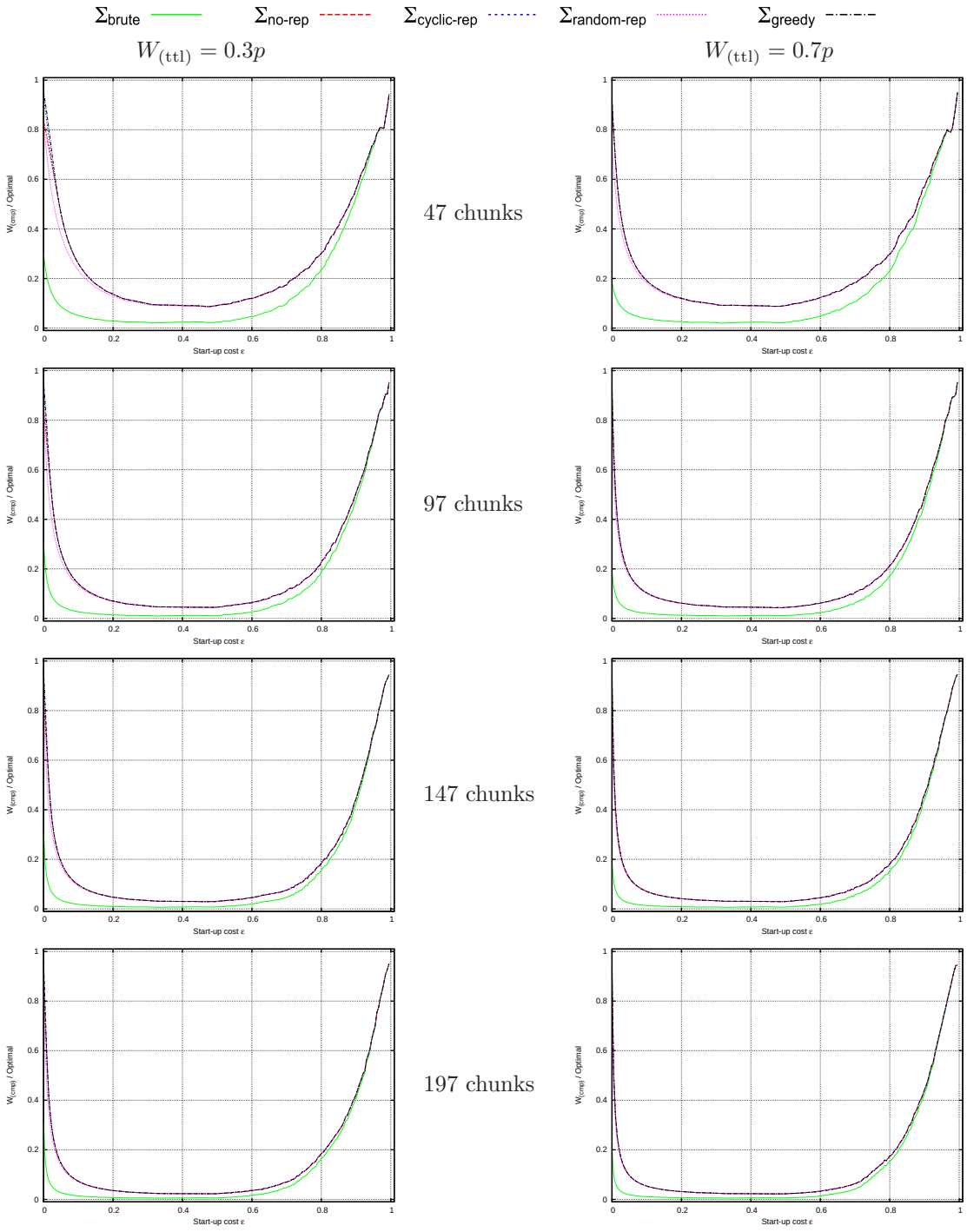


Figure 34: Experiment (E4) with 10 computers.

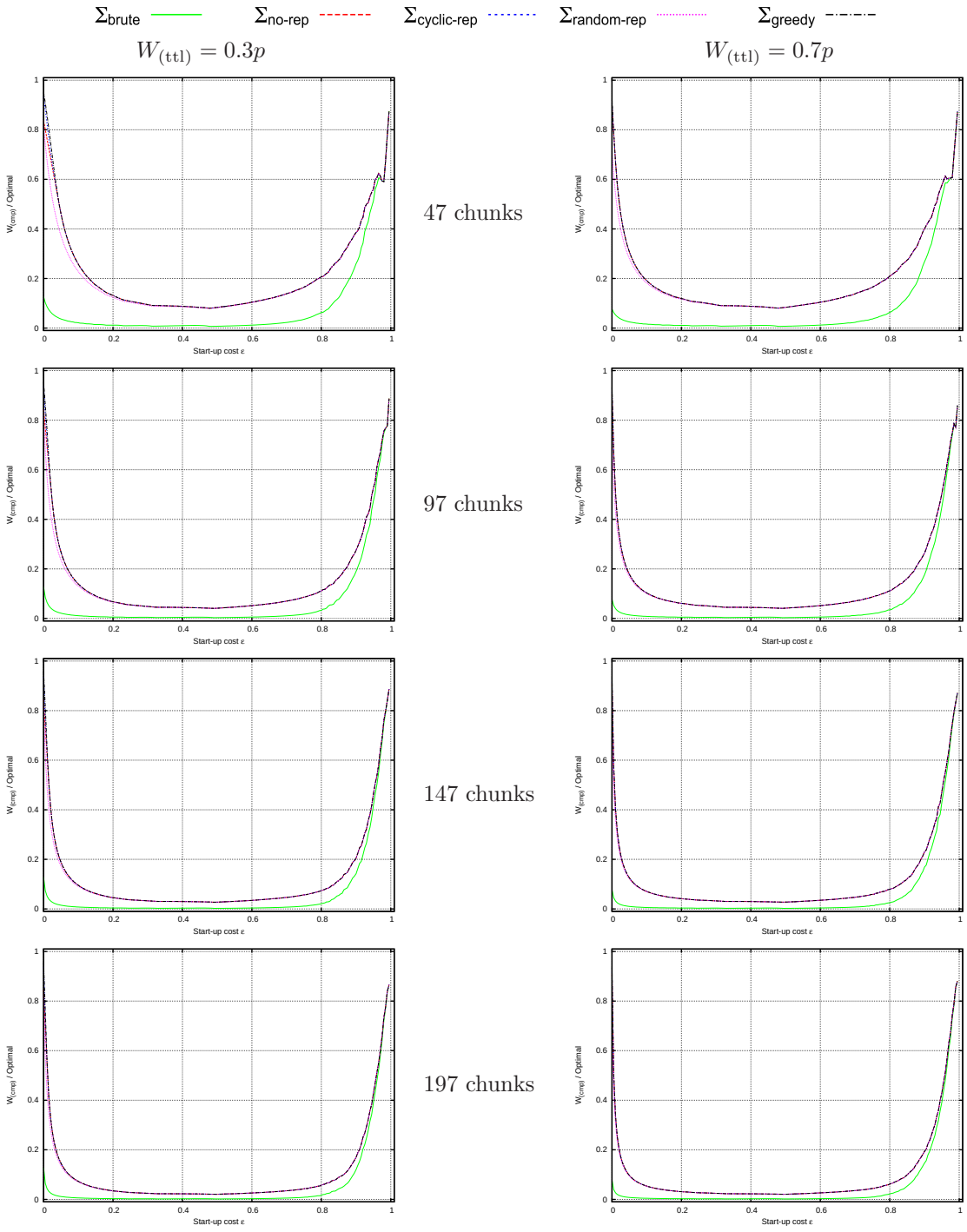


Figure 35: Experiment (E4) with 25 computers.

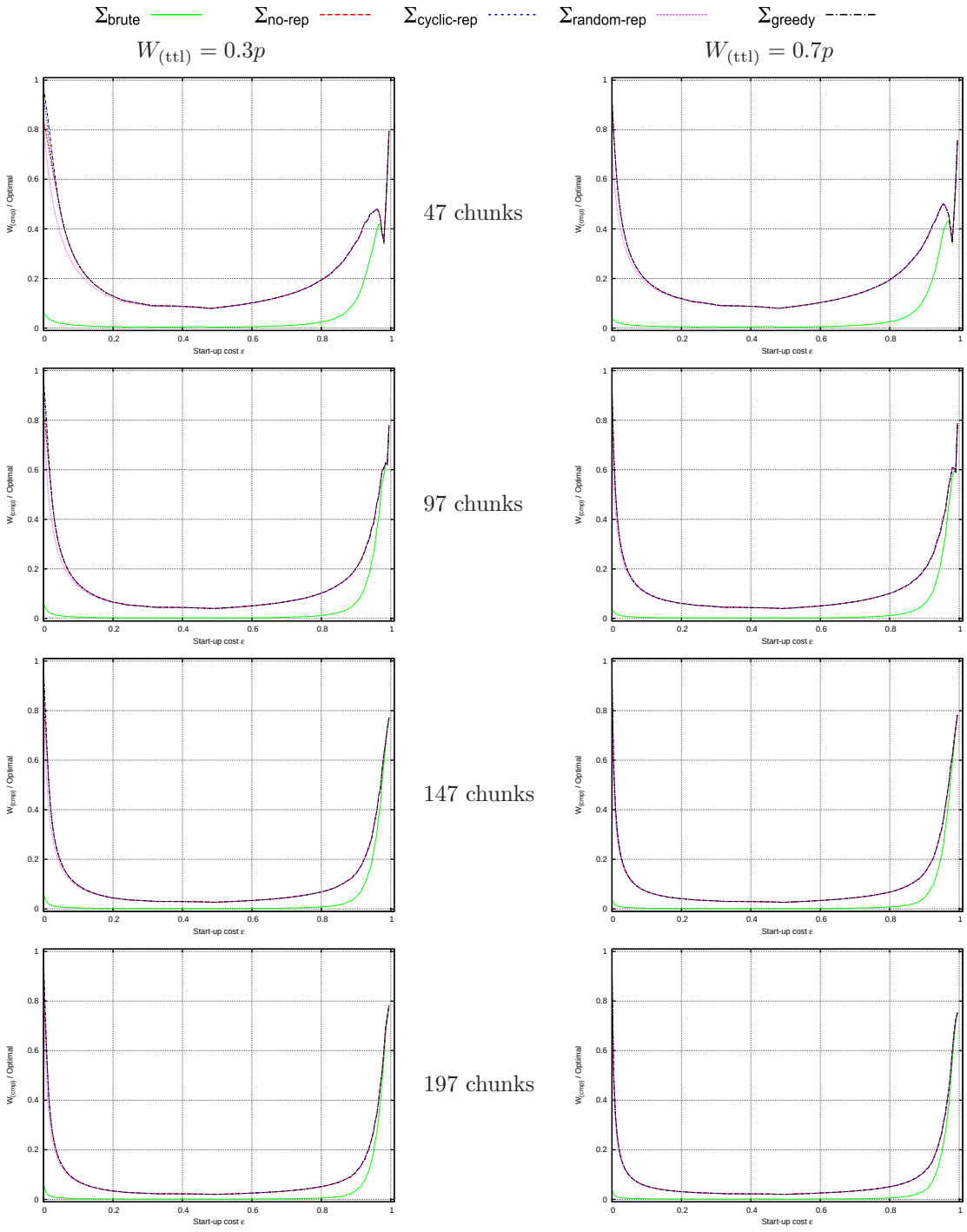


Figure 36: Experiment (E4) with 50 computers.

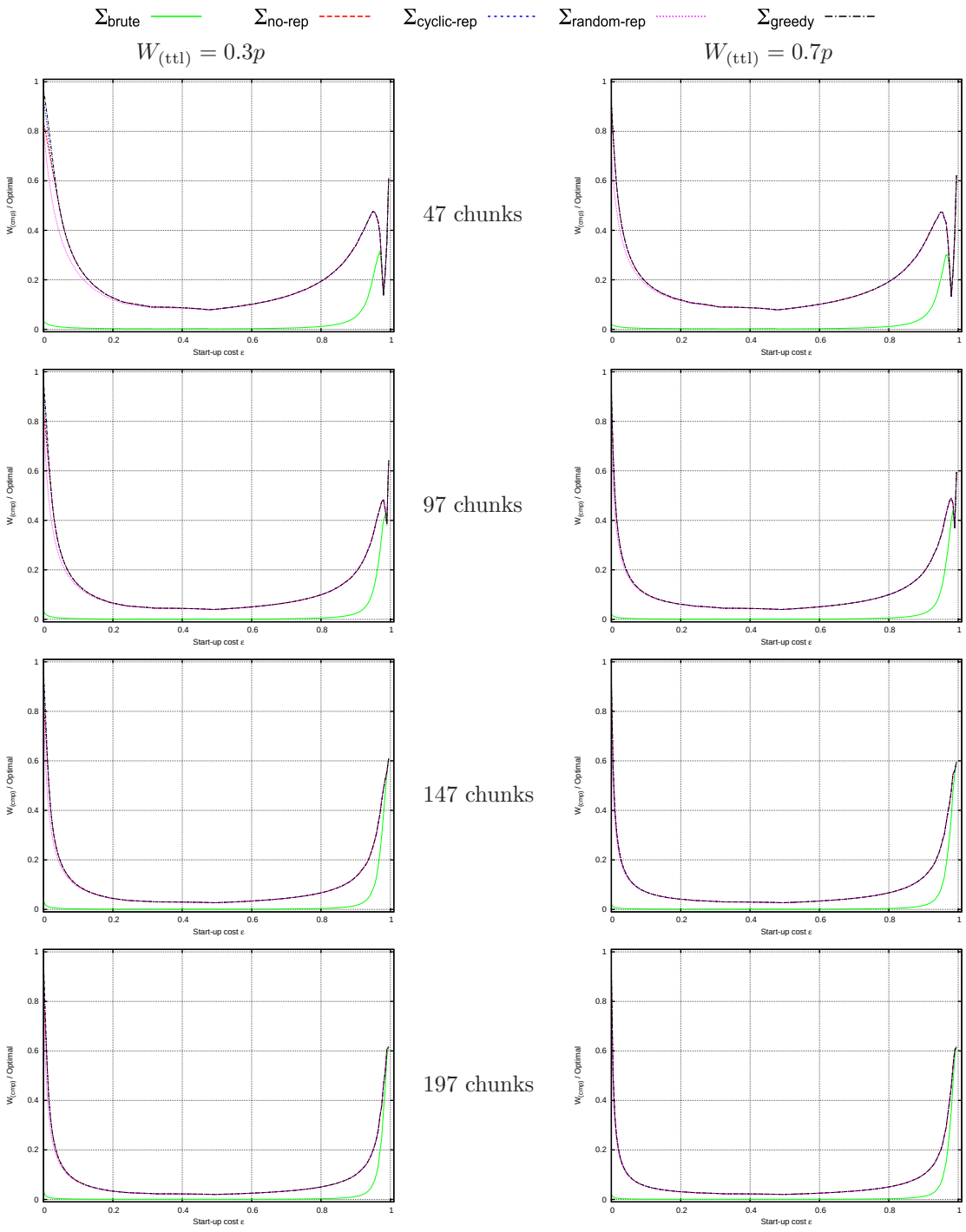


Figure 37: Experiment (E4) with 100 computers.

B Experiments with linear risk functions (all heuristics)

On the following graphs, the performance of all the heuristics is displayed, including all our group heuristics.

B.1 Experiments E1

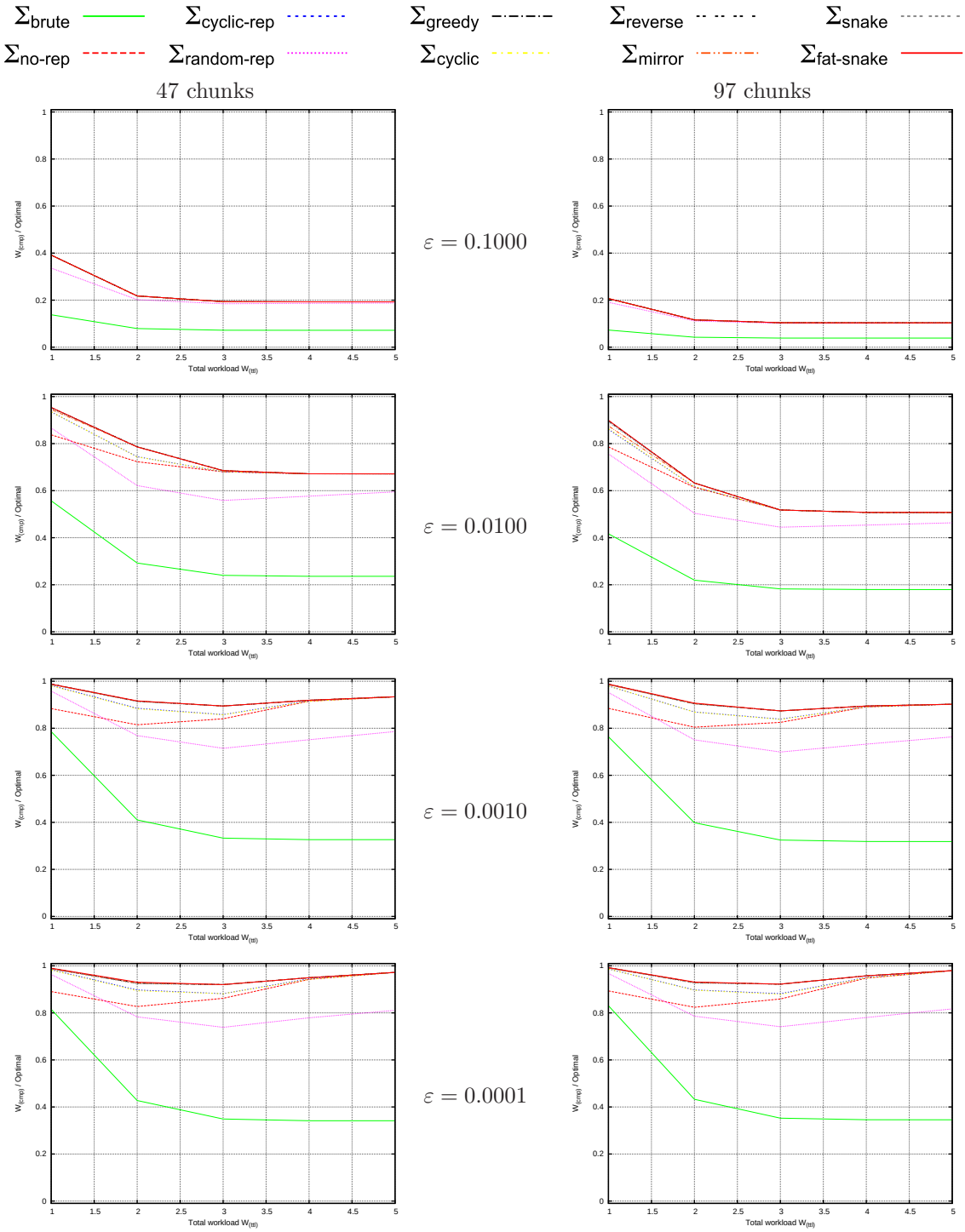


Figure 38: Experiment (E1) using 5 computers.

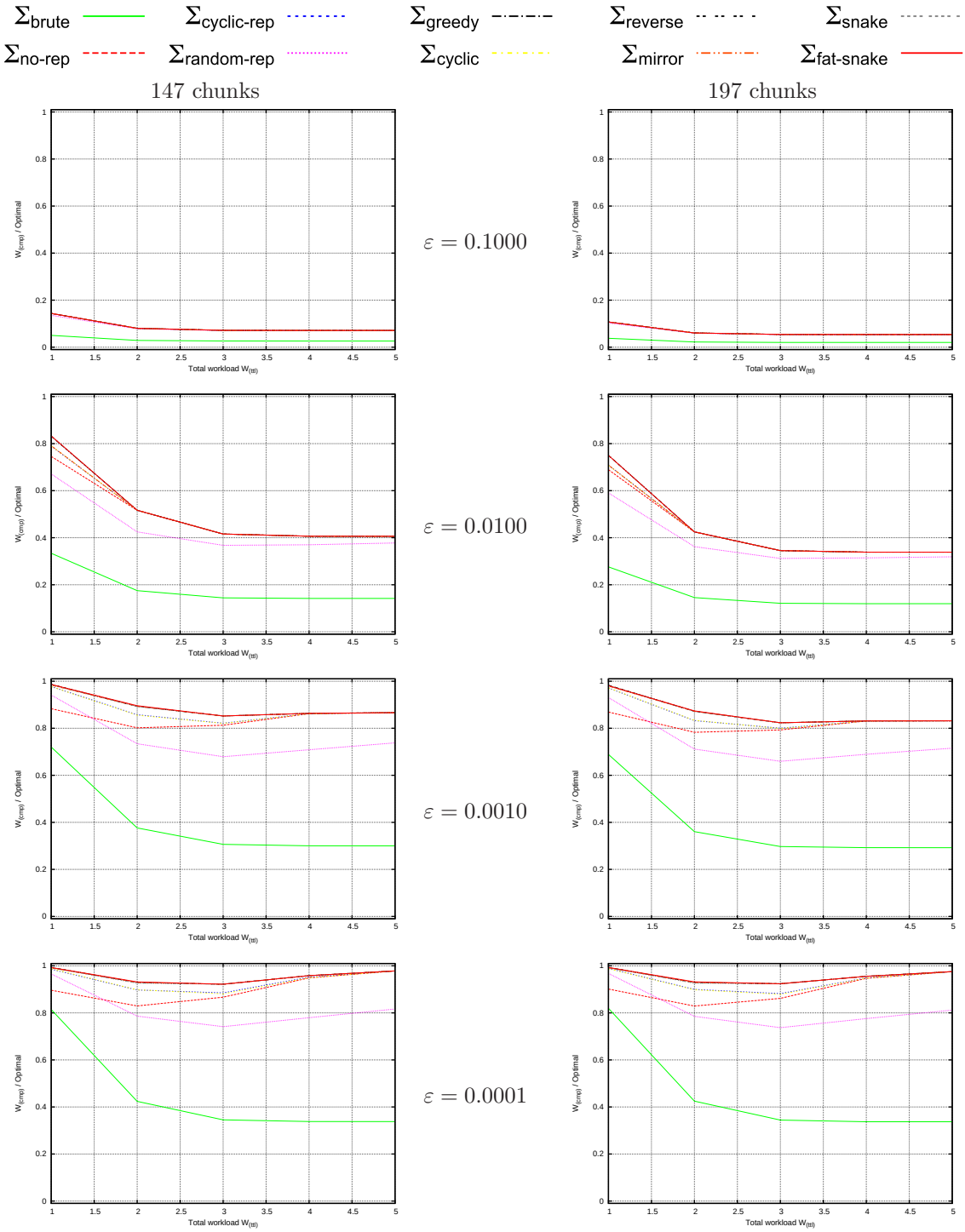


Figure 39: Experiment (E1) using 5 computers (continued).

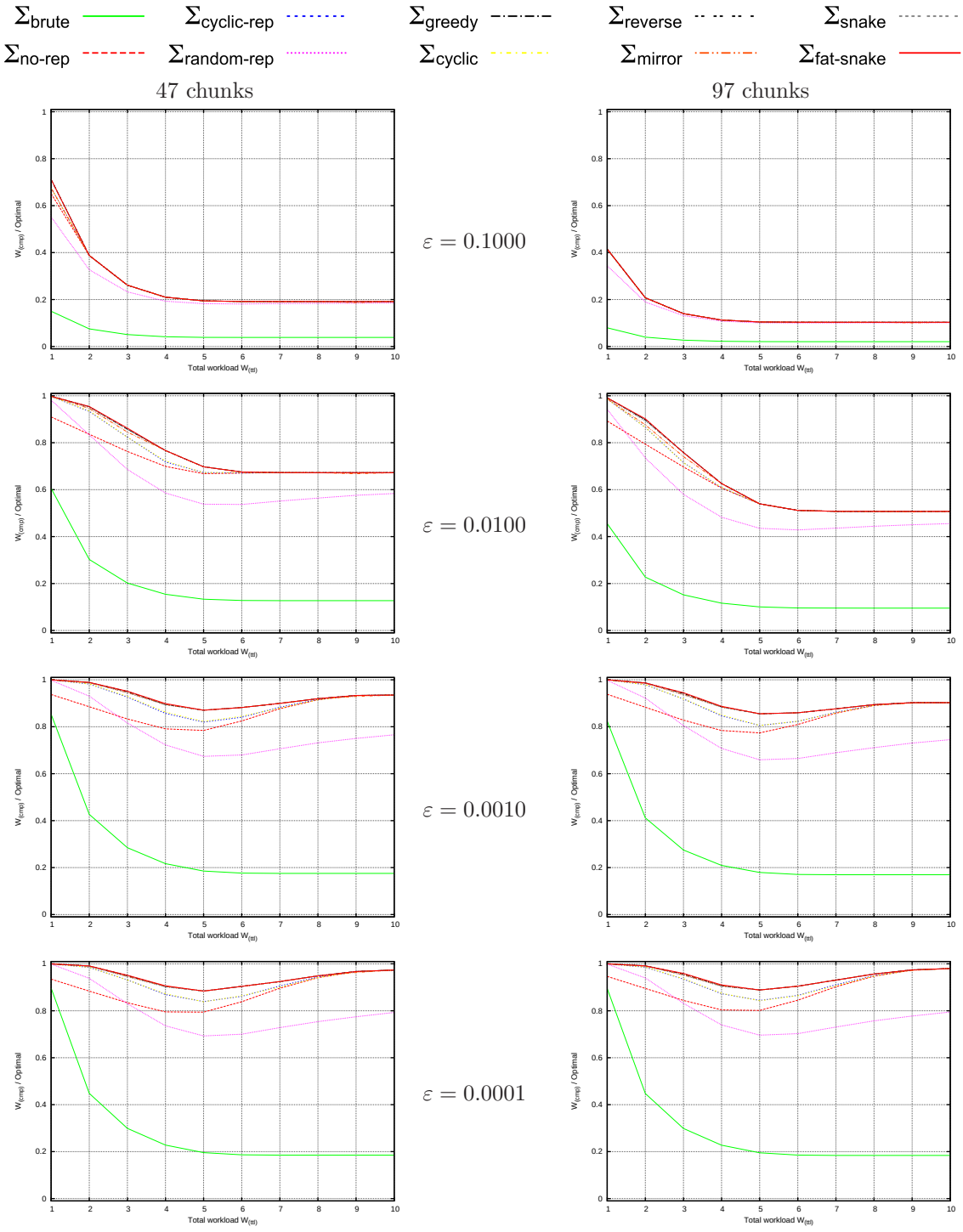


Figure 40: Experiment (E1) using 10 computers.

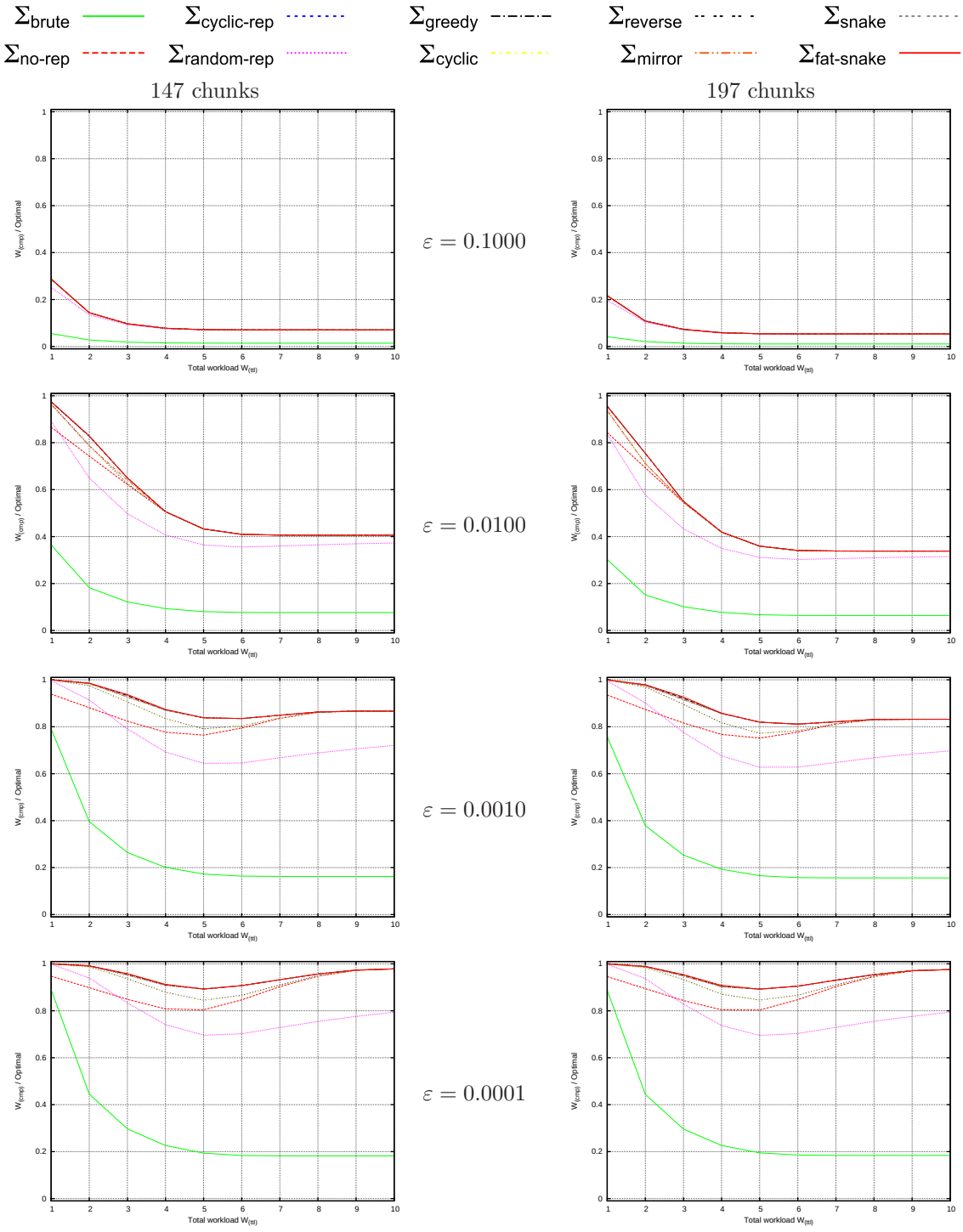


Figure 41: Experiment (E1) using 10 computers (continued).

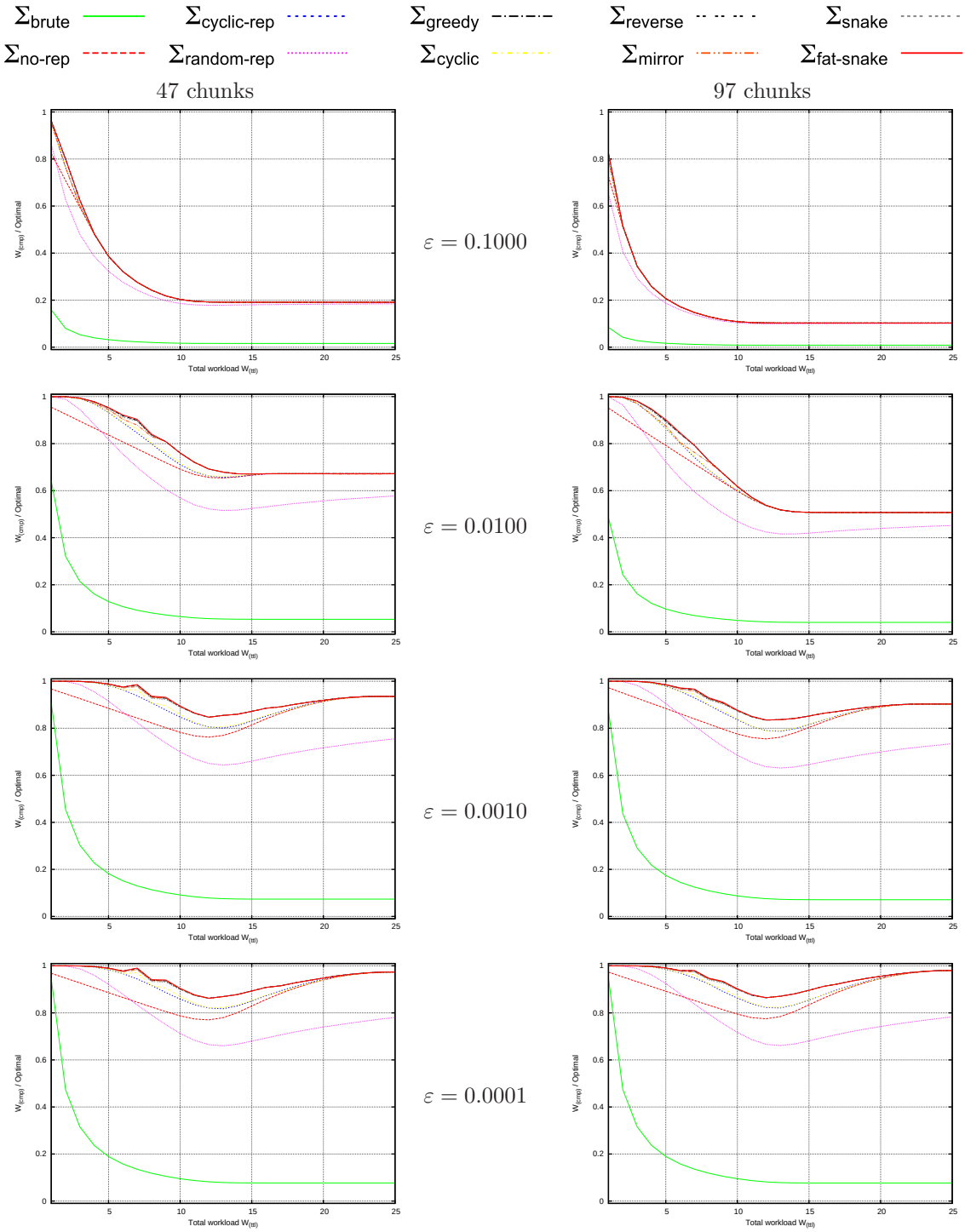


Figure 42: Experiment (E1) using 25 computers.

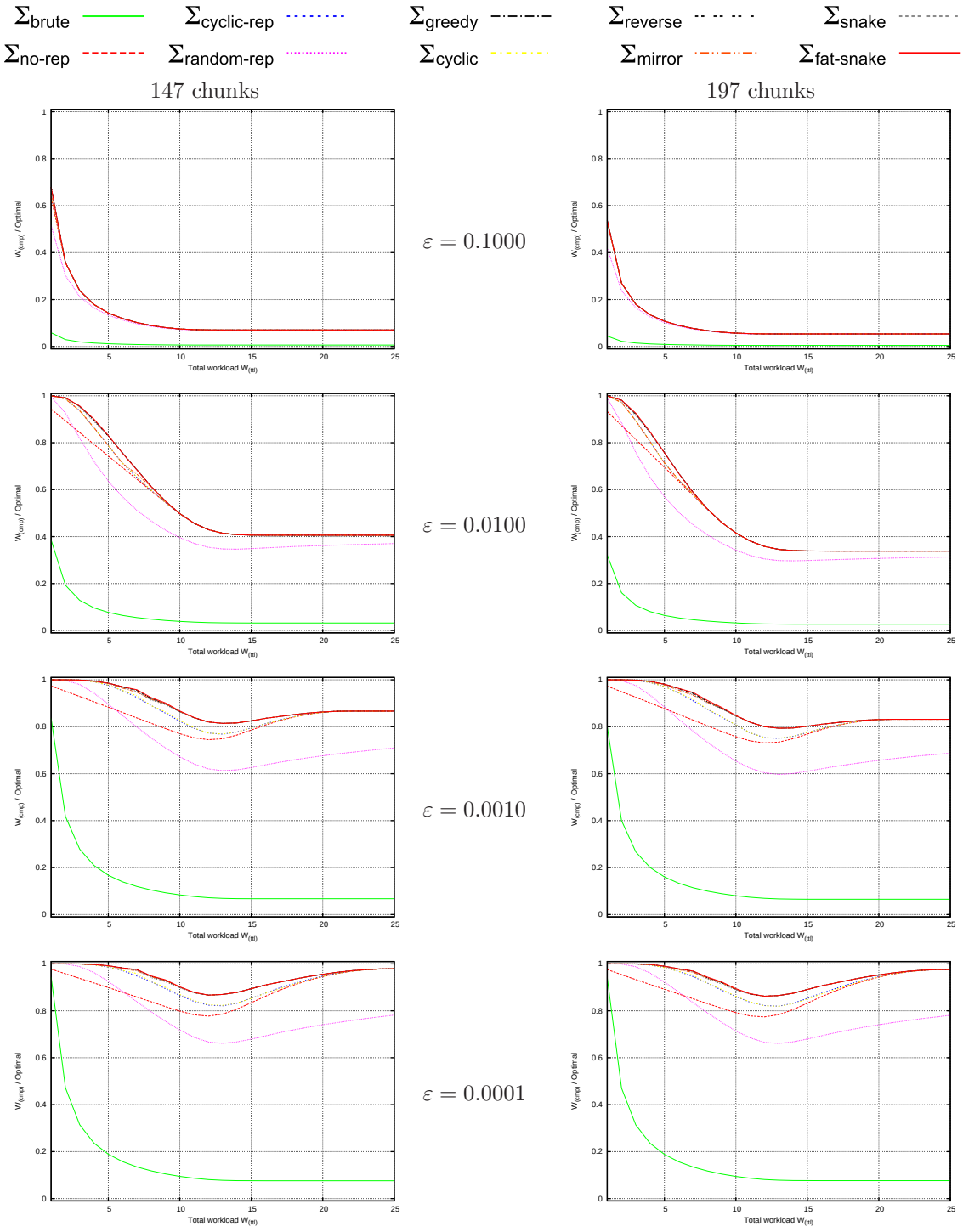


Figure 43: Experiment (E1) using 25 computers (continued).

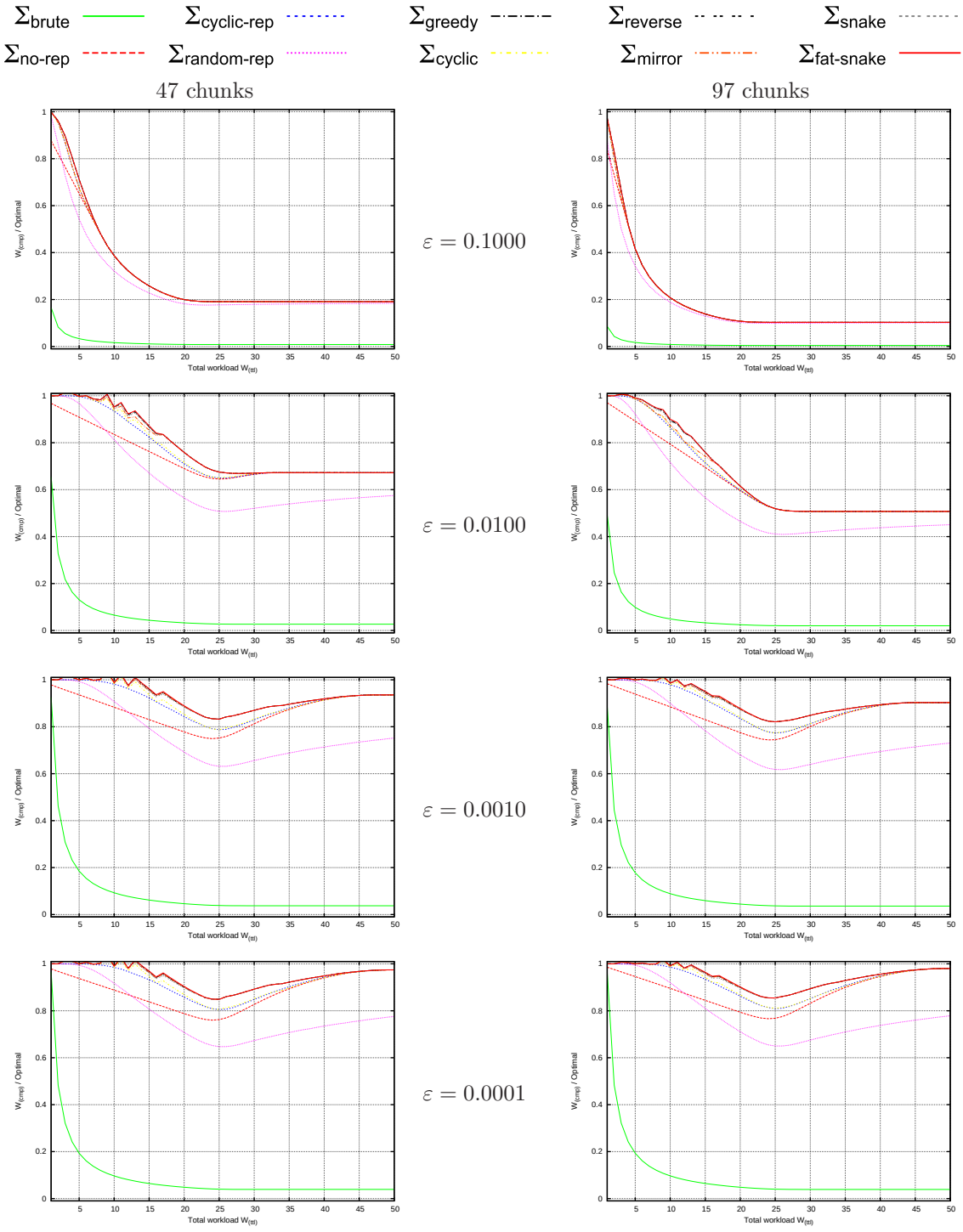


Figure 44: Experiment (E1) using 50 computers.

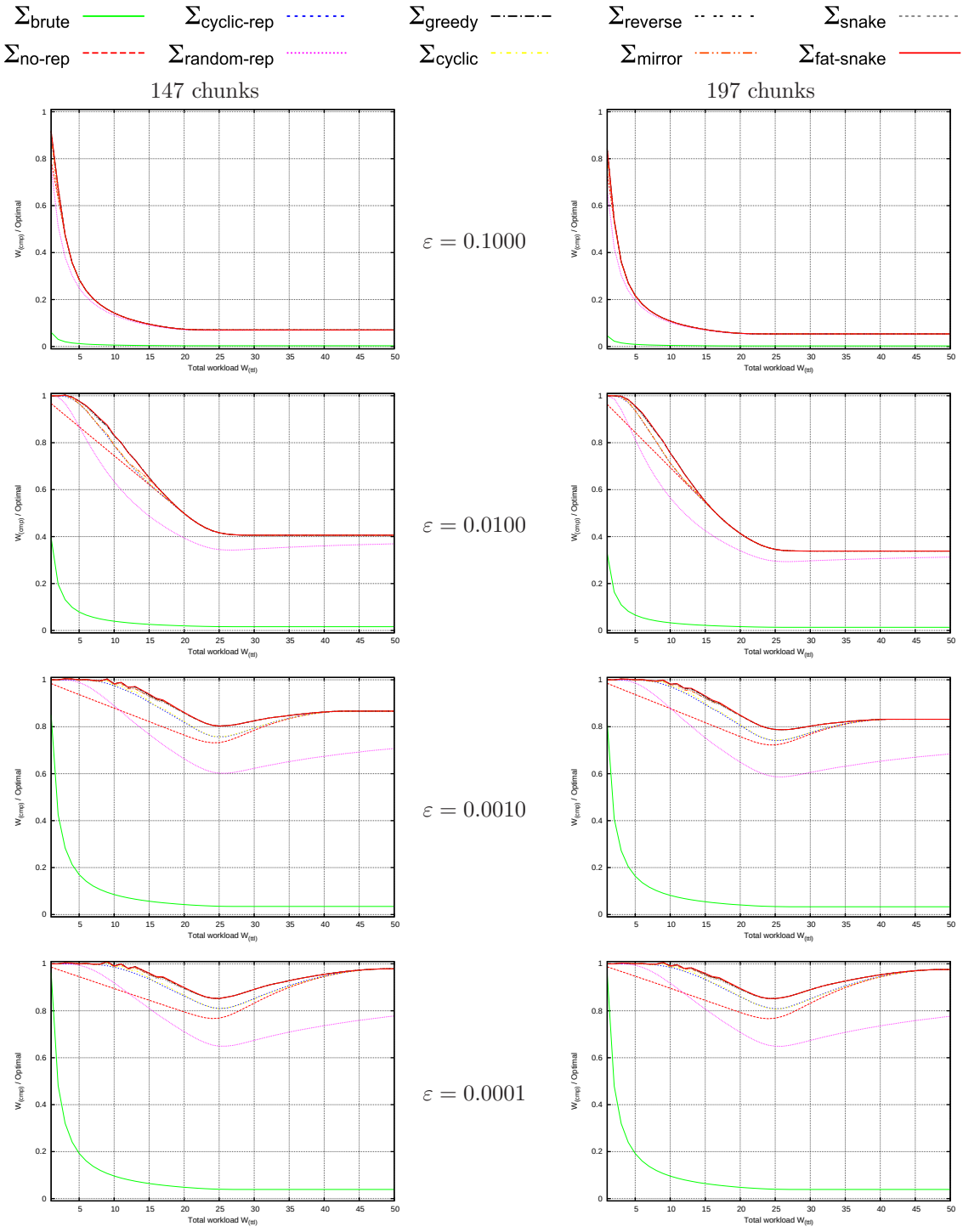


Figure 45: Experiment (E1) using 50 computers (continued).

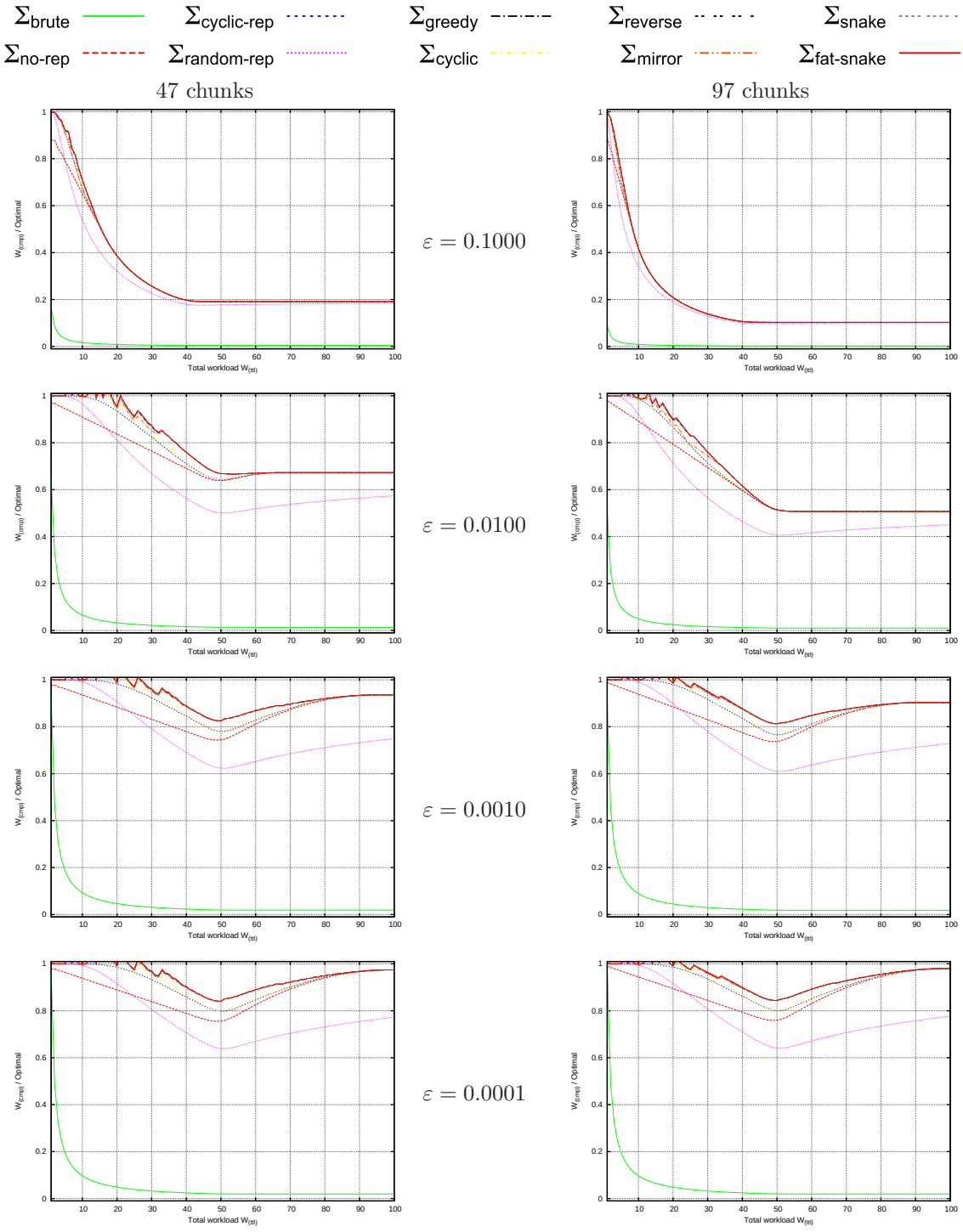


Figure 46: Experiment (E1) using 100 computers.

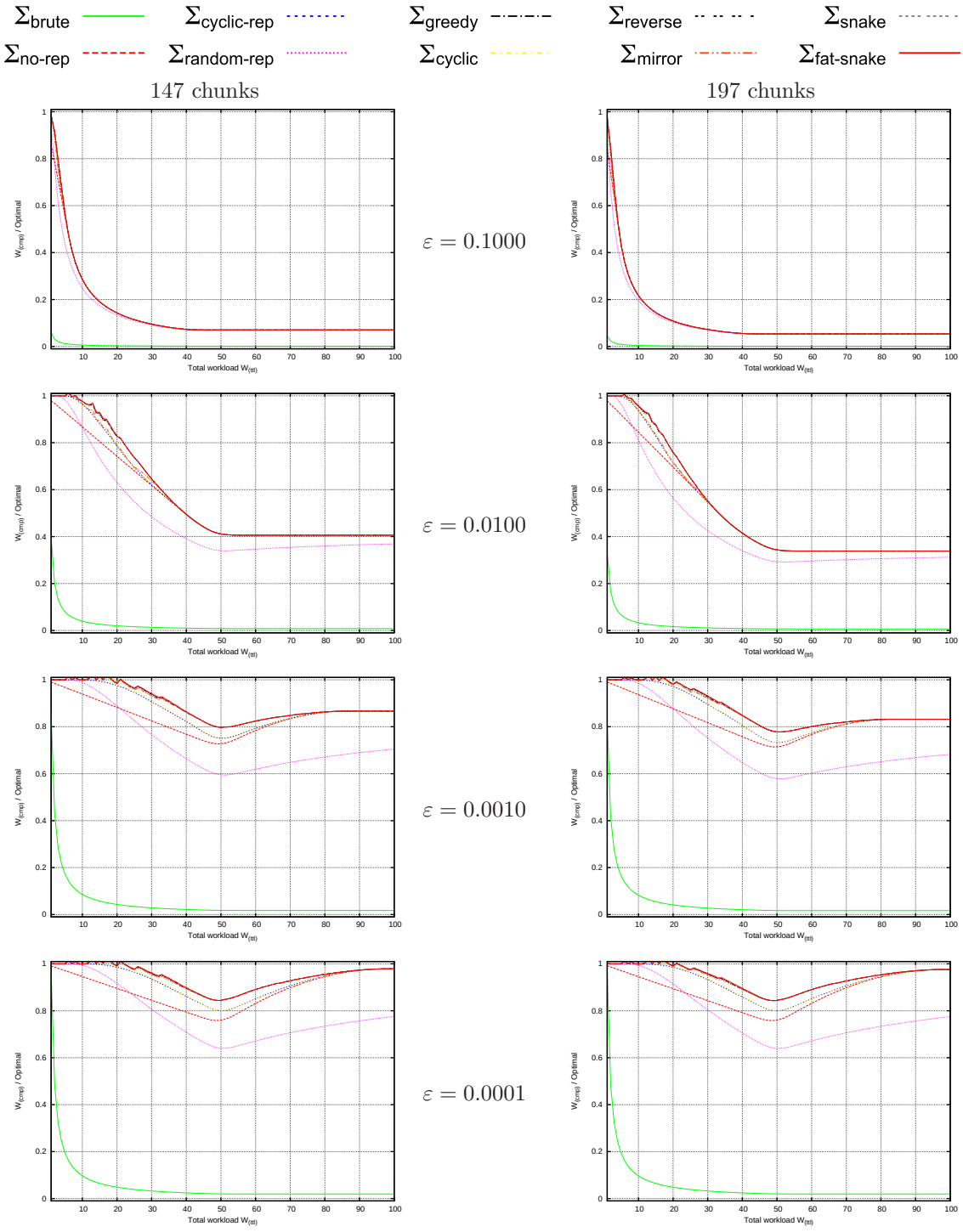


Figure 47: Experiment (E1) using 100 computers (continued).

B.2 Experiments E2

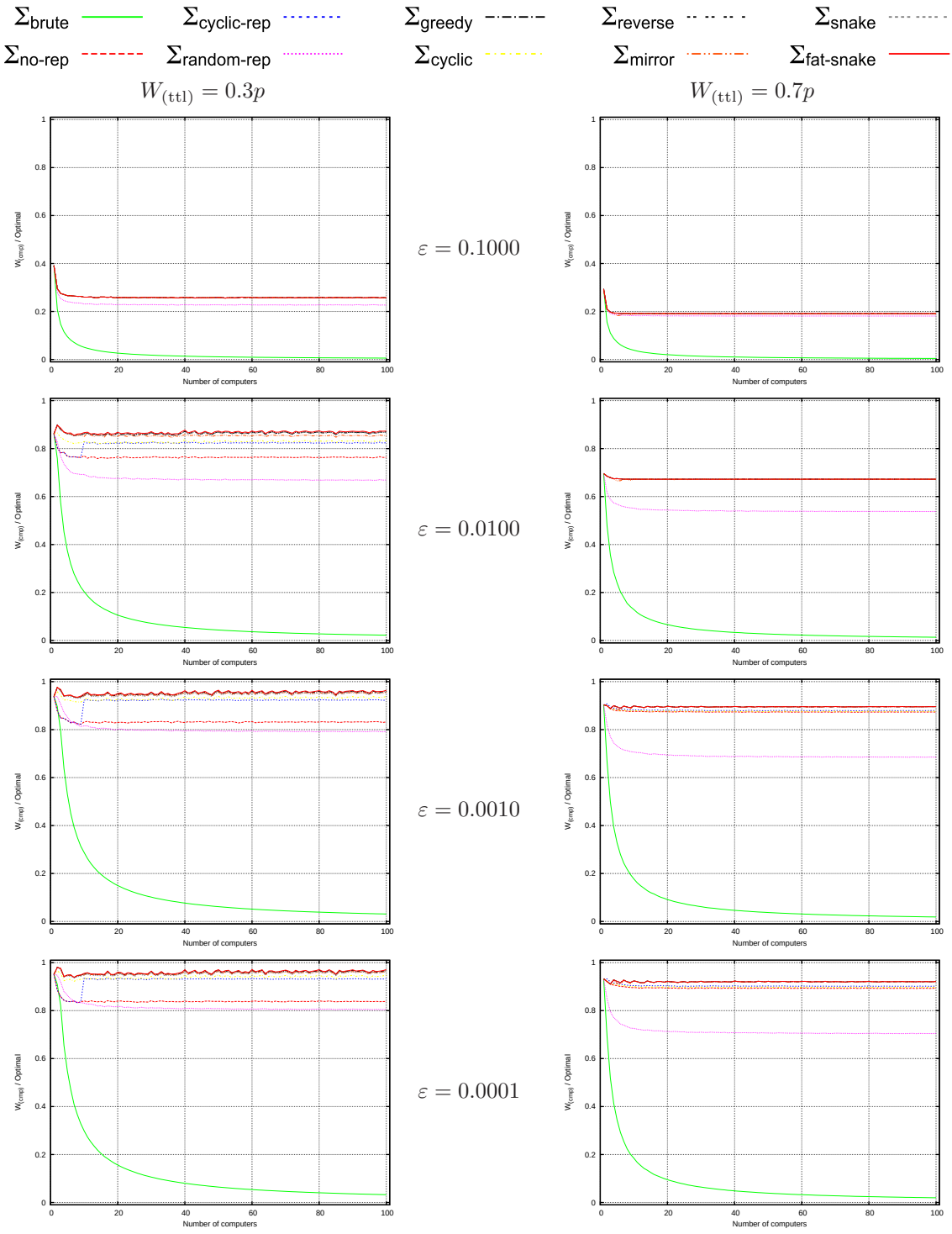


Figure 48: Experiment (E2) with 47 chunks.

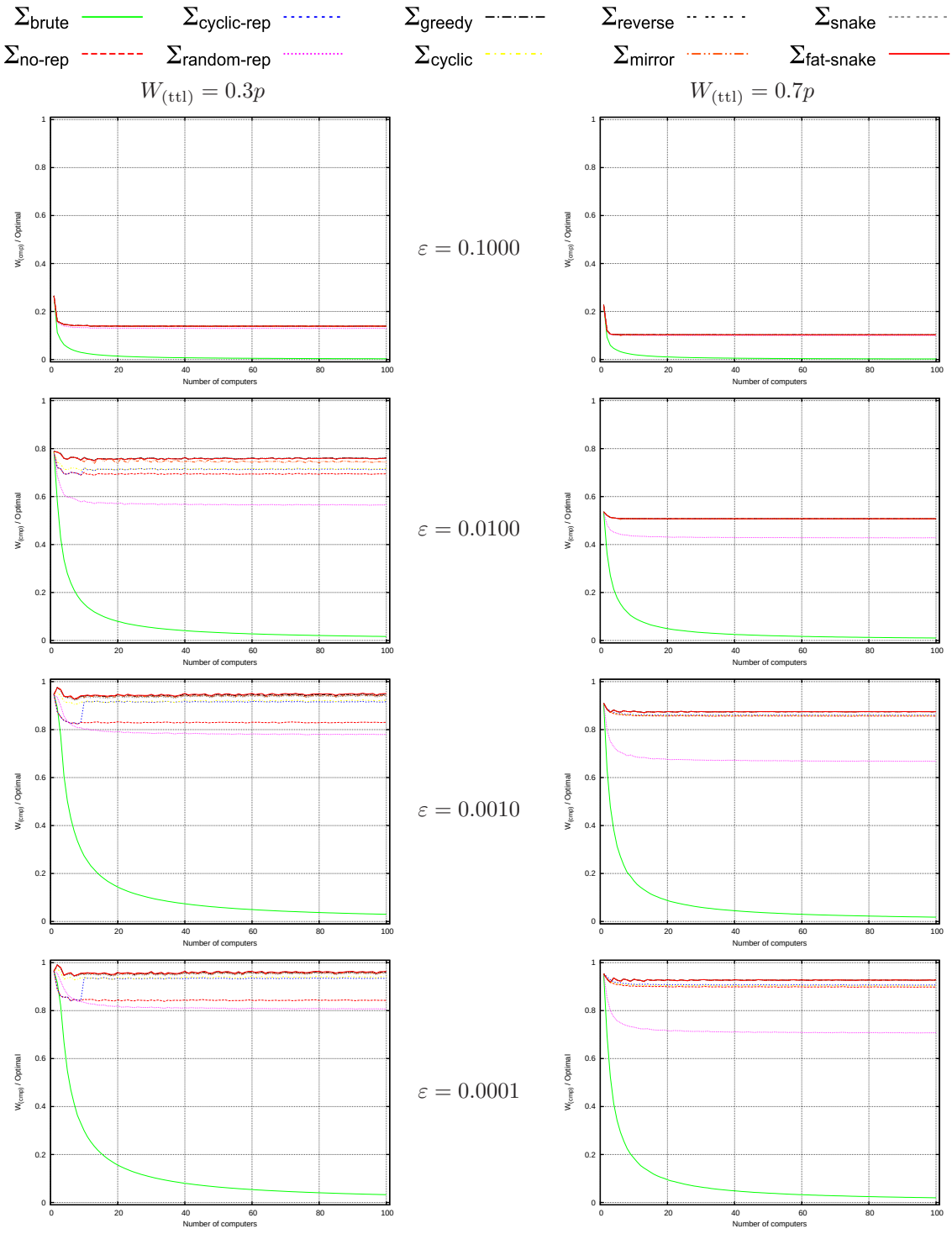


Figure 49: Experiment (E2) with 97 chunks.

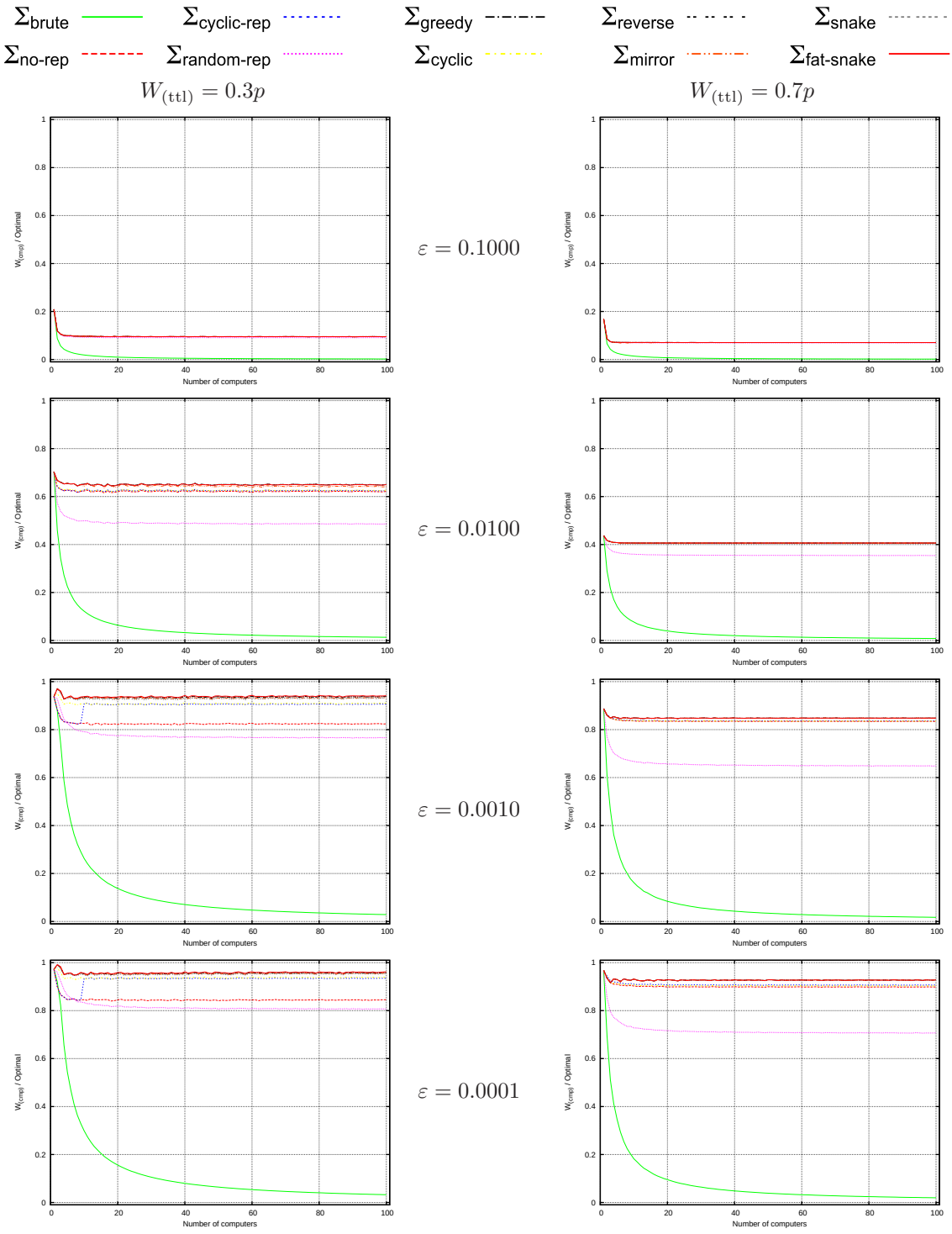


Figure 50: Experiment (E2) with 147 chunks.

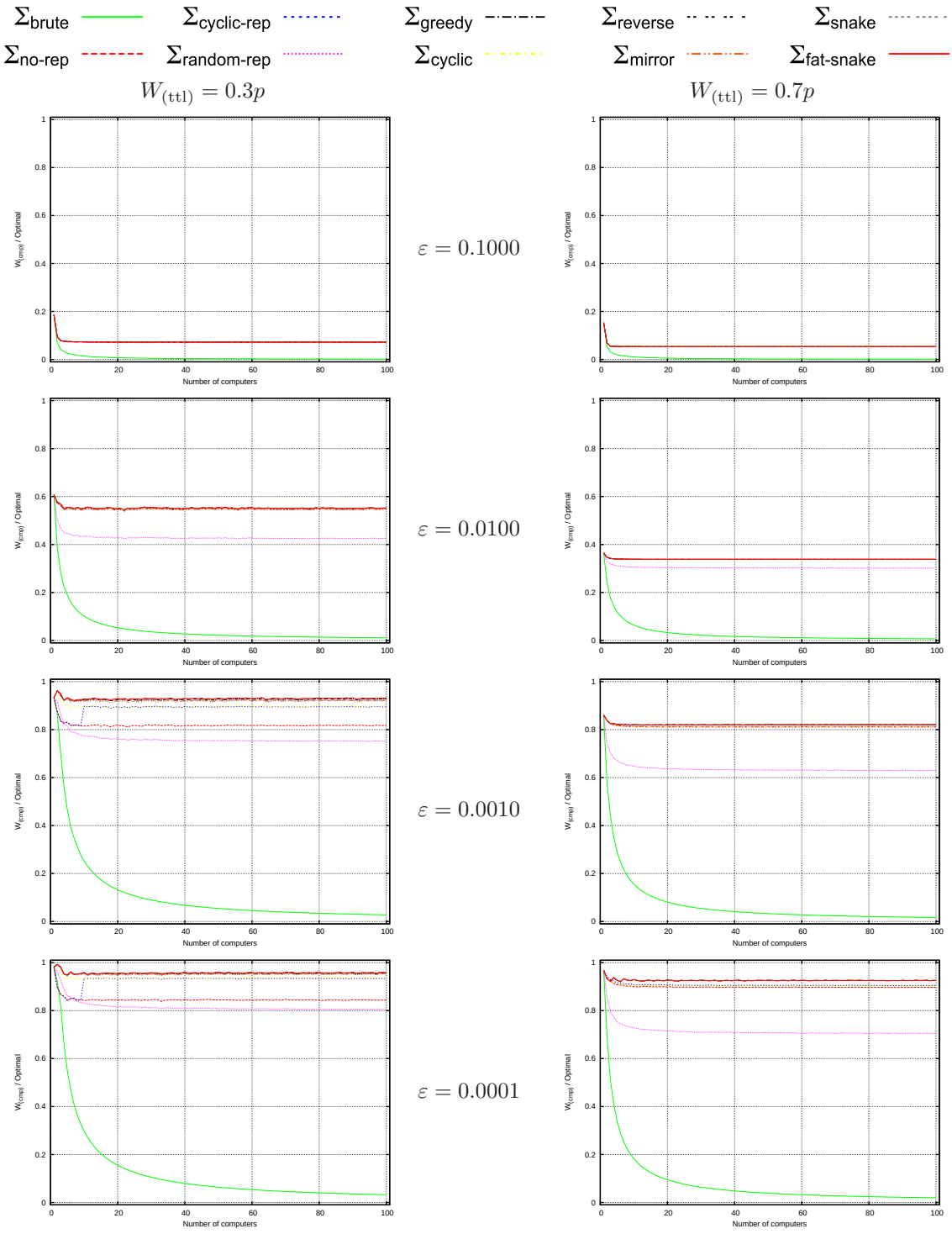


Figure 51: Experiment (E2) with 197 chunks.

B.3 Experiments E3

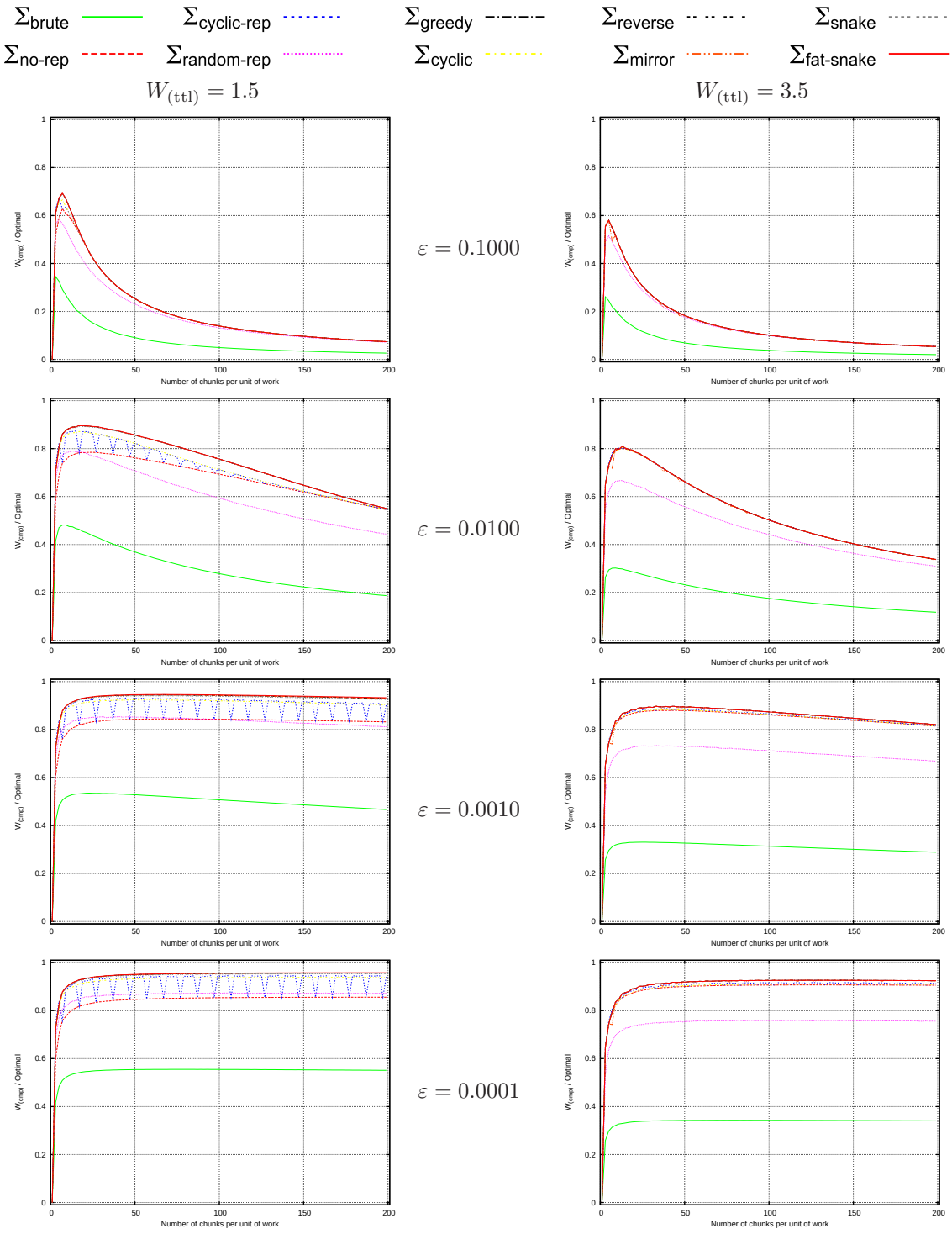


Figure 52: Experiment (E3) using 5 computers.

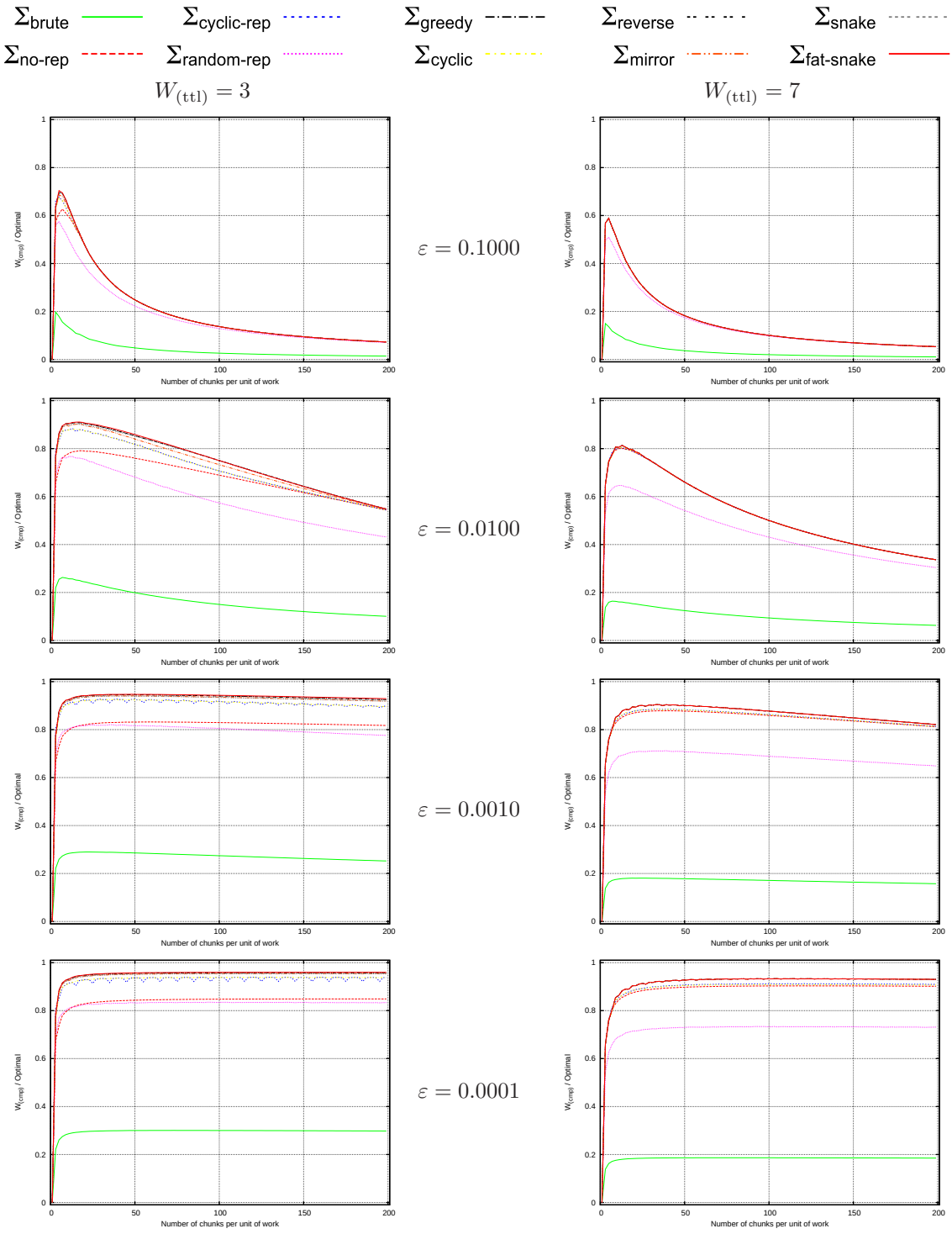


Figure 53: Experiment (E3) using 10 computers.

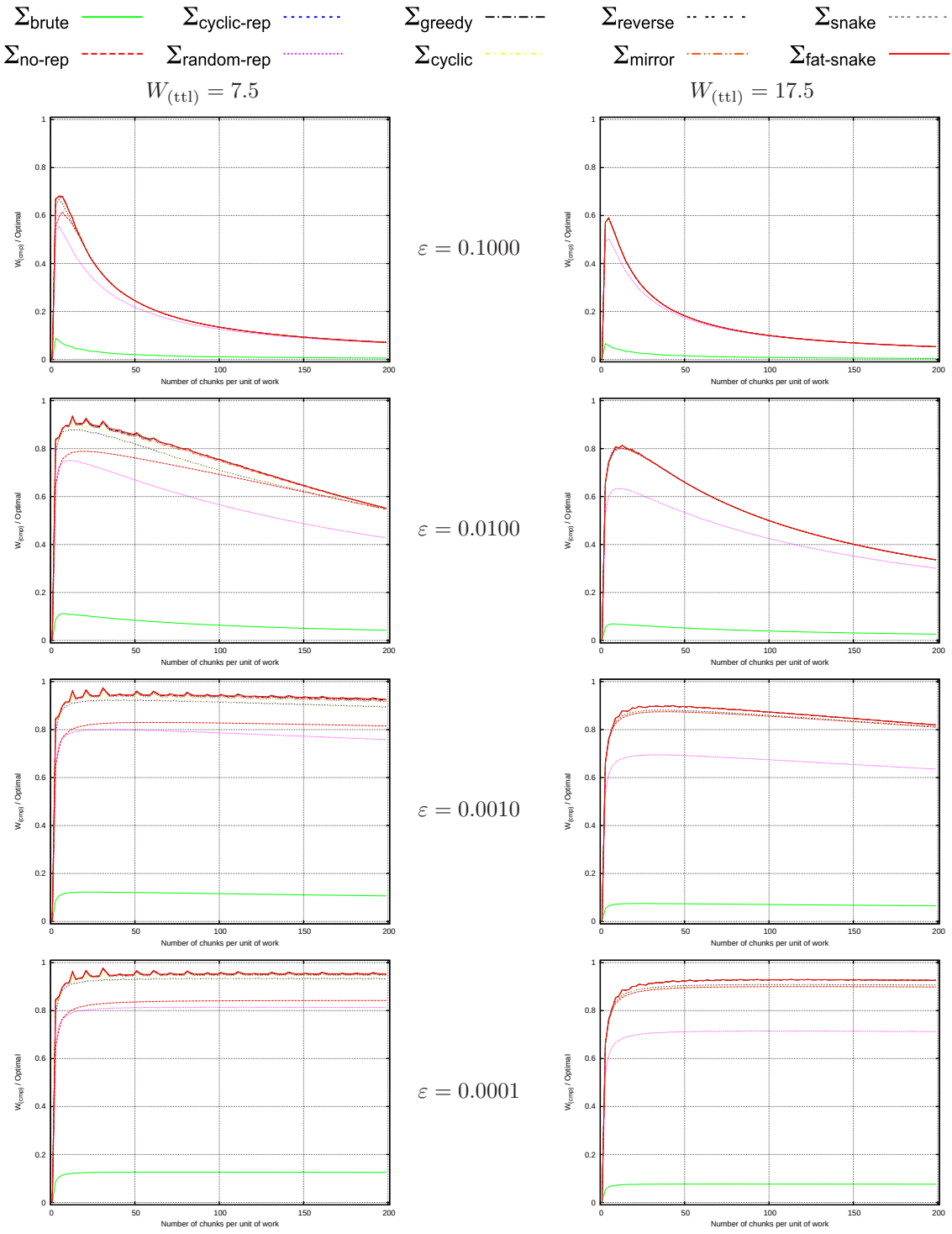


Figure 54: Experiment (E3) using 25 computers.

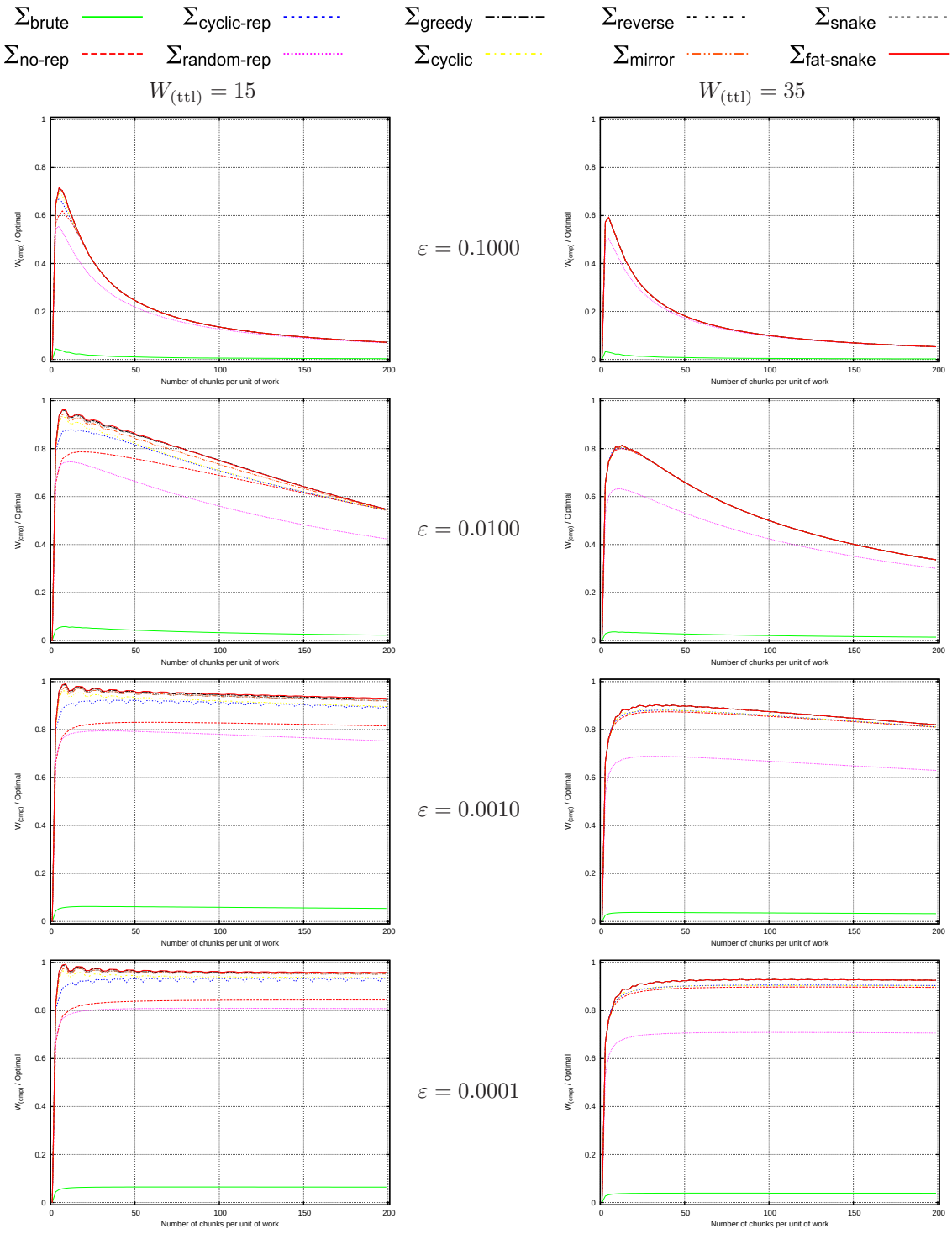


Figure 55: Experiment (E3) using 50 computers.

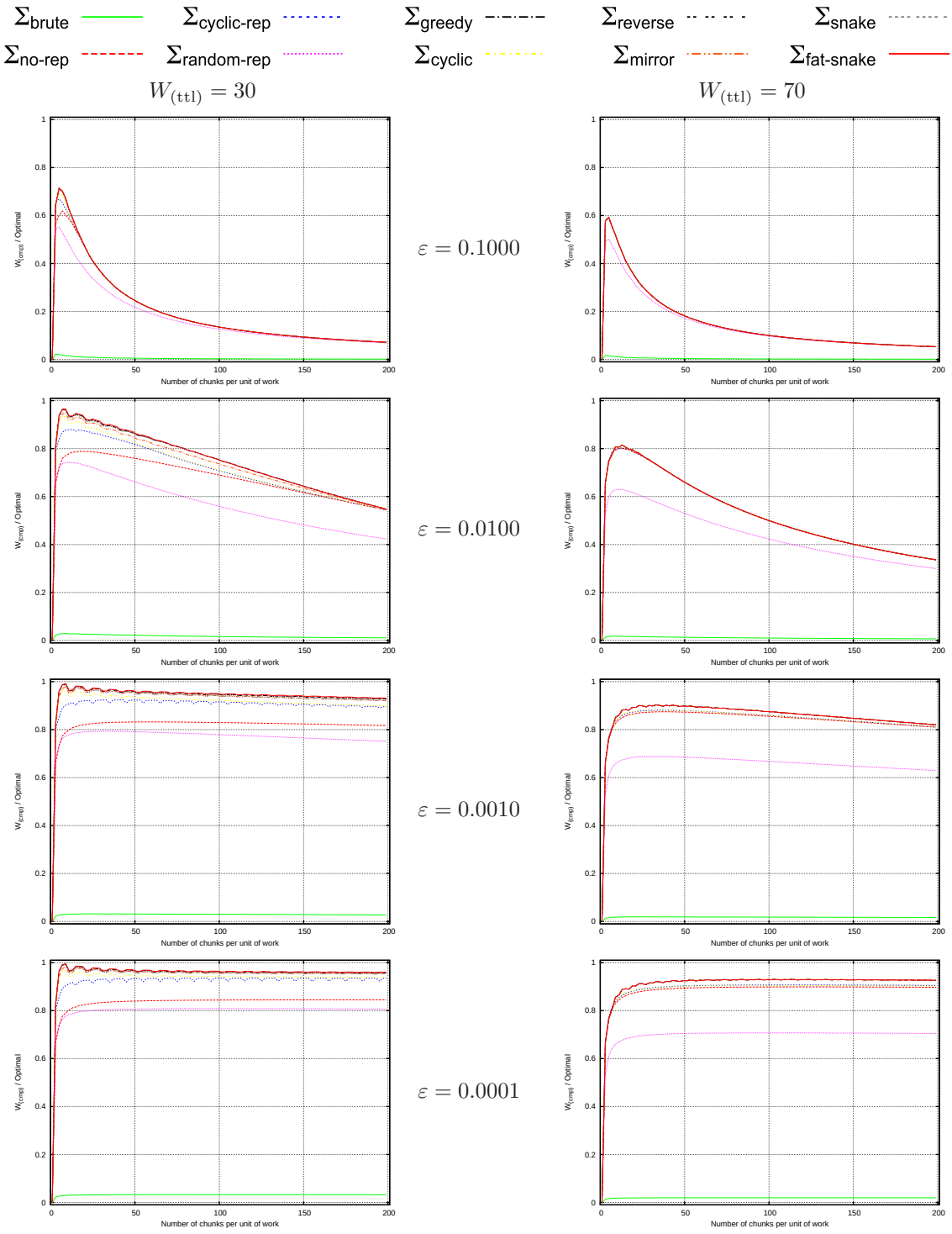


Figure 56: Experiment (E3) using 100 computers.

B.4 Experiments E4

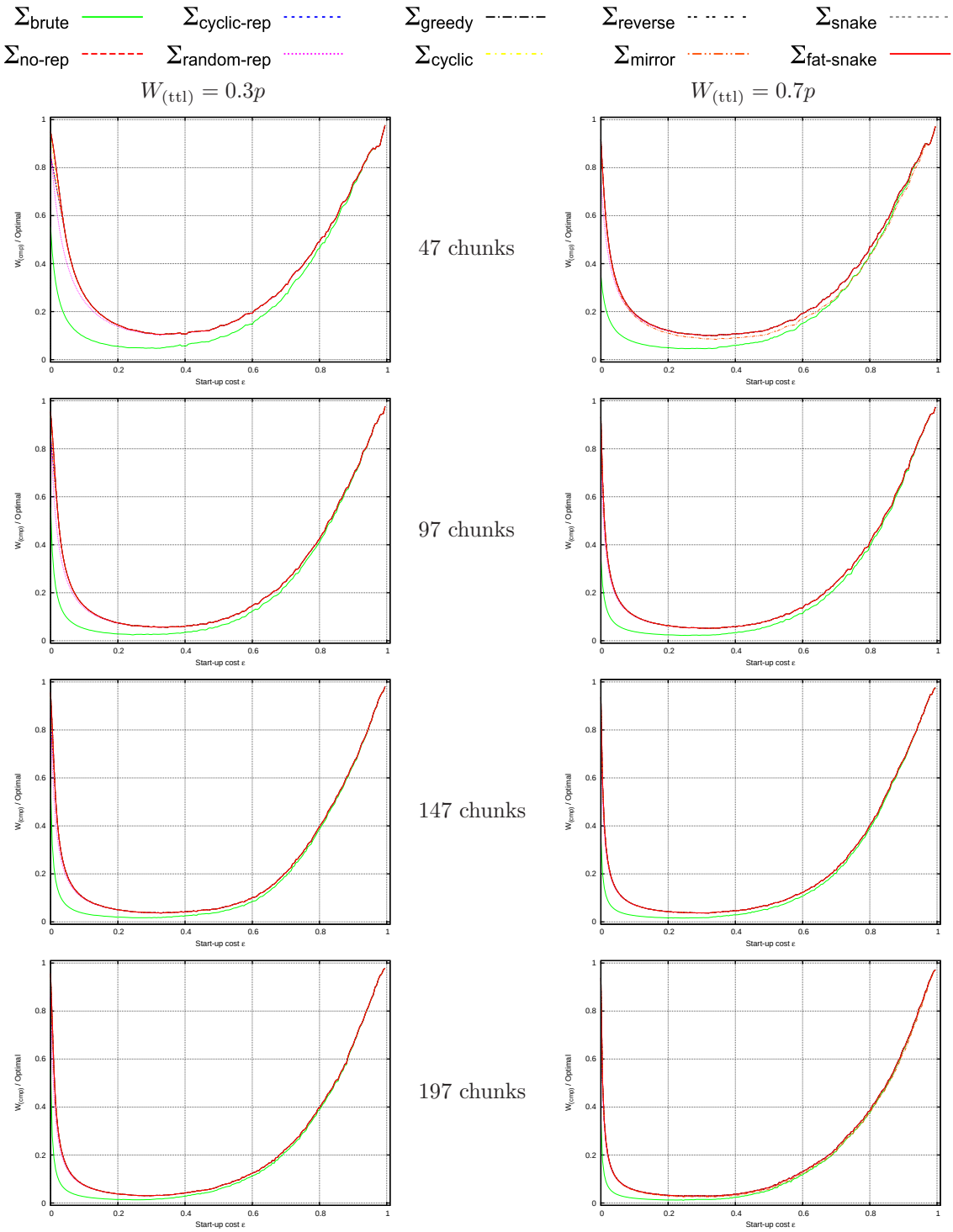


Figure 57: Experiment (E4) with 5 computers.

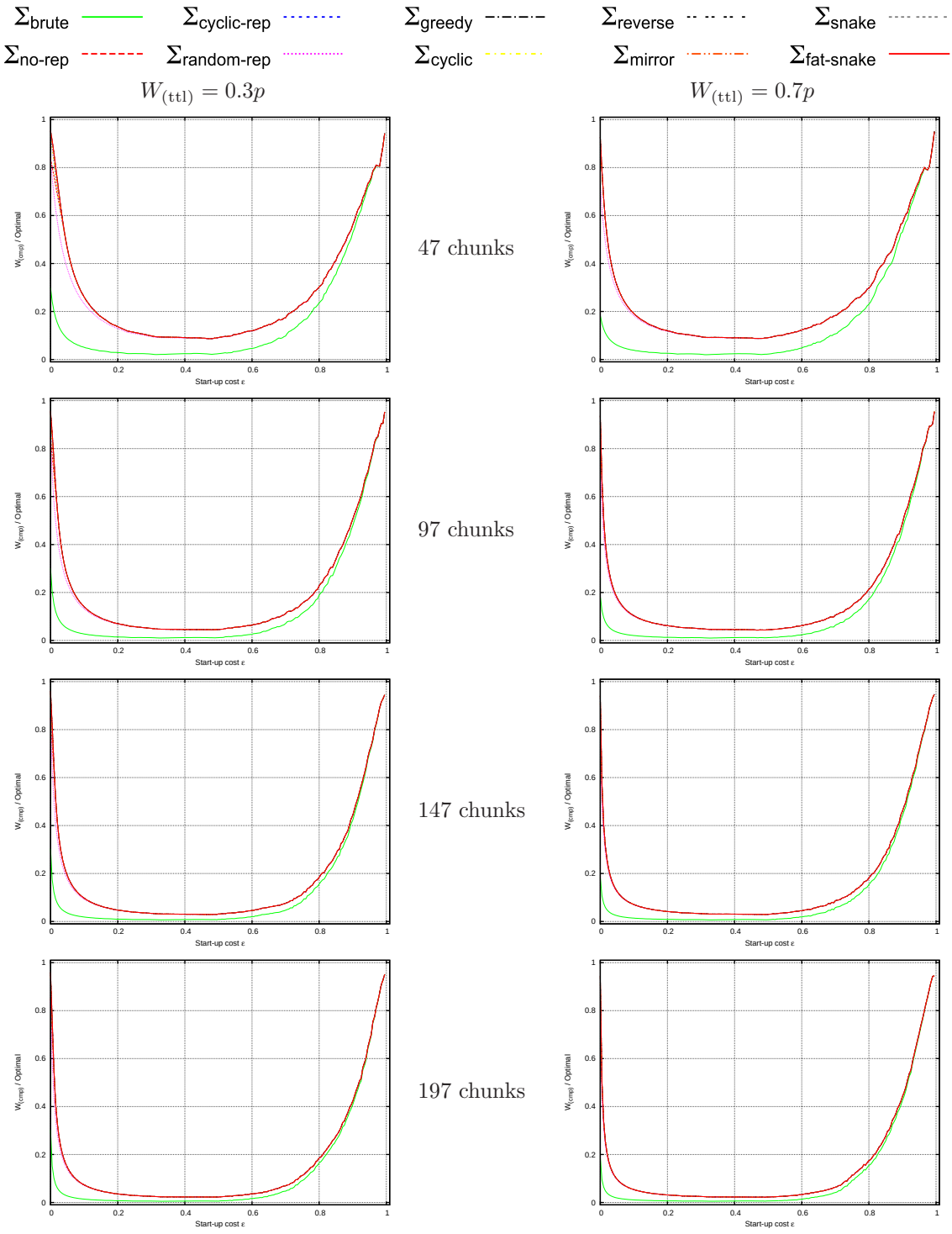


Figure 58: Experiment (E4) with 10 computers.

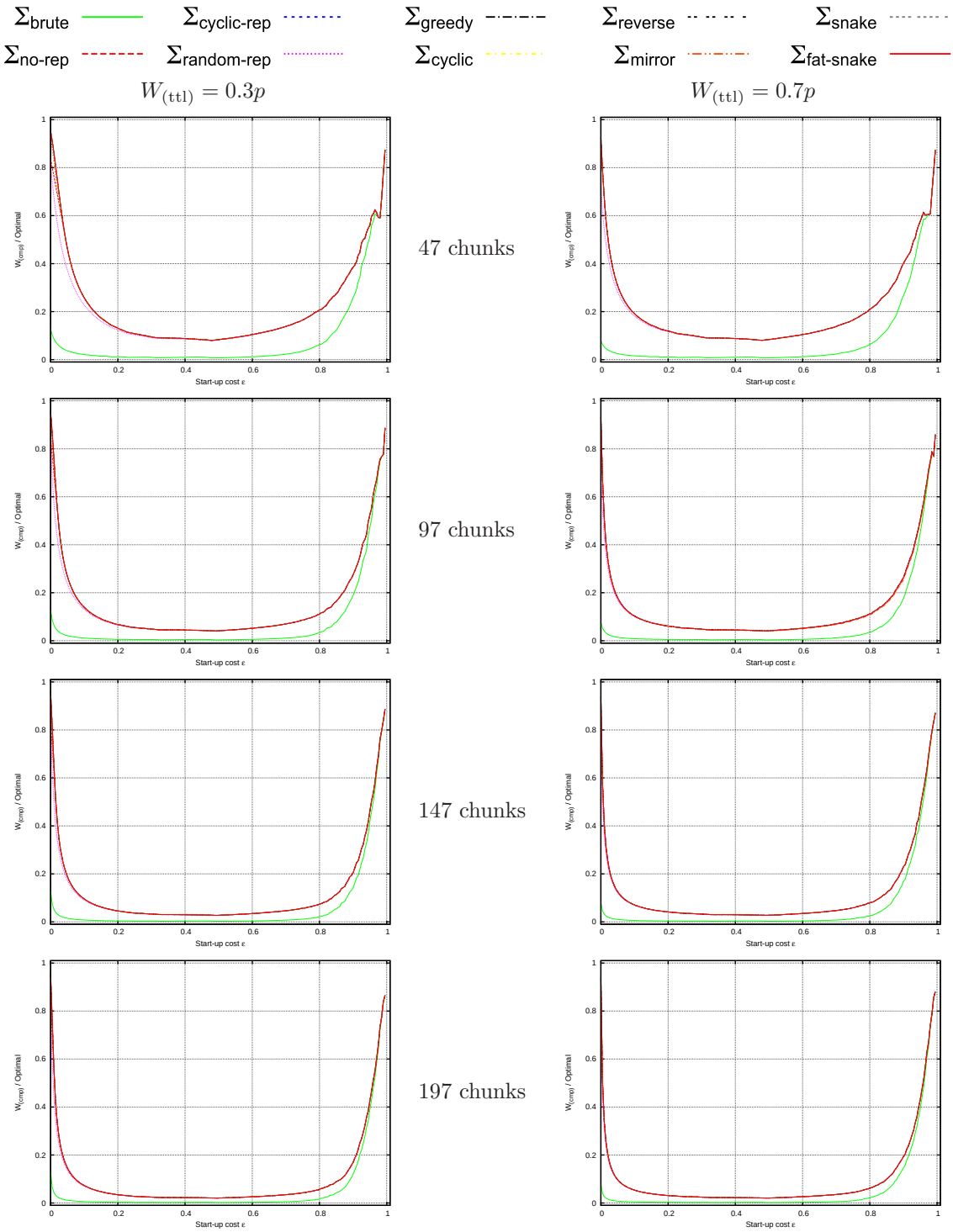


Figure 59: Experiment (E4) with 25 computers.

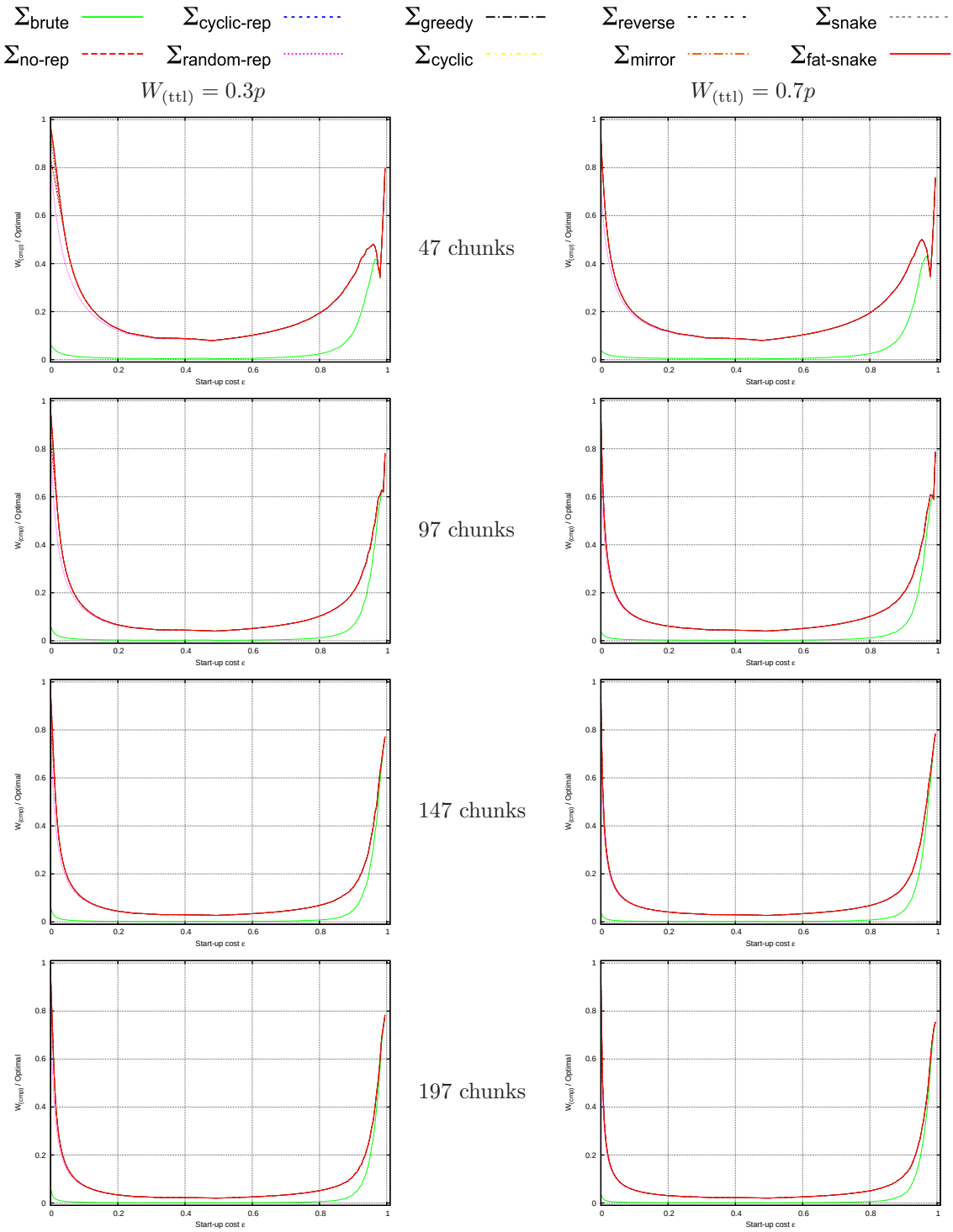


Figure 60: Experiment (E4) with 50 computers.

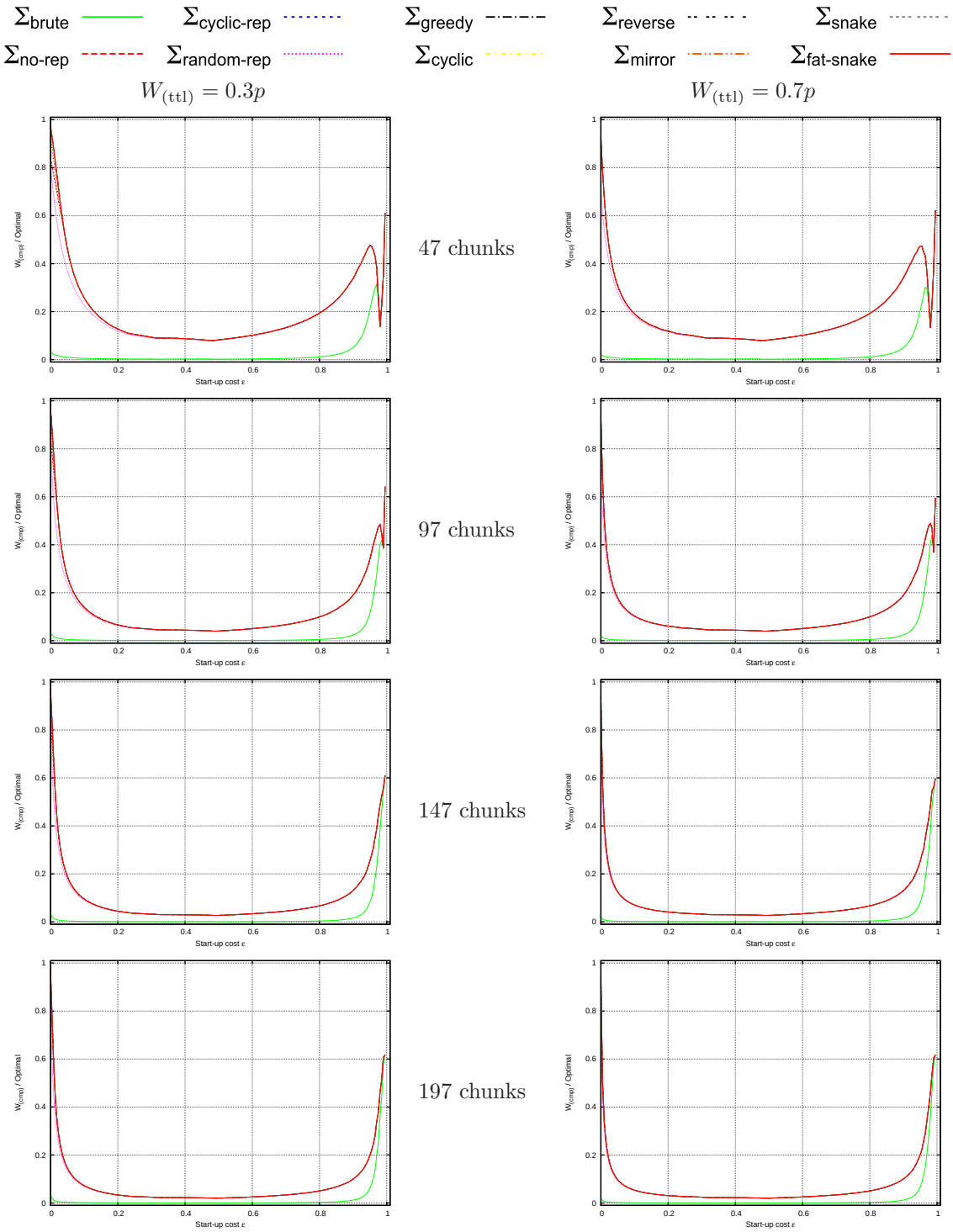
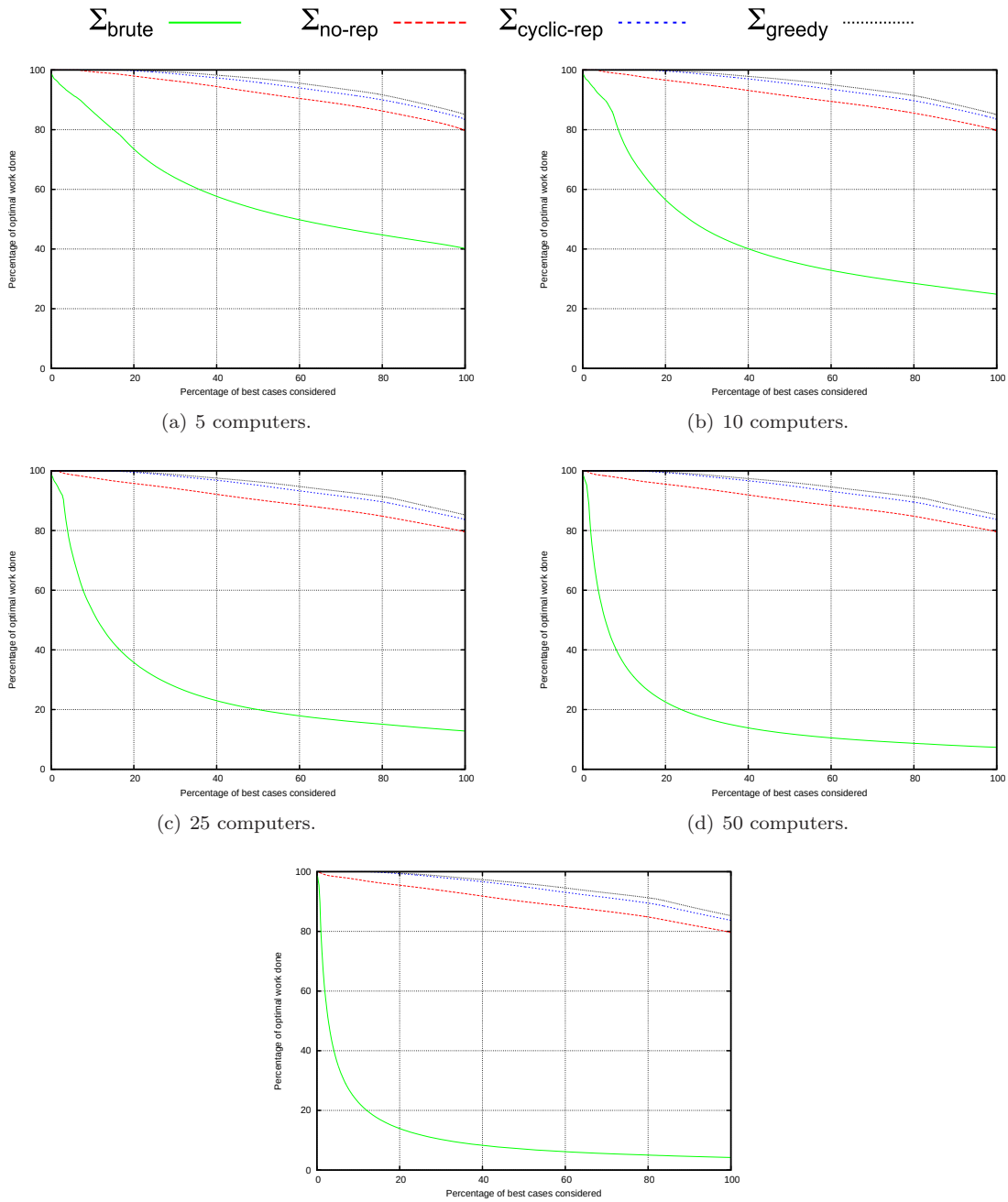


Figure 61: Experiment (E4) with 100 computers.

B.5 Experiments E5



(a) 5 computers.

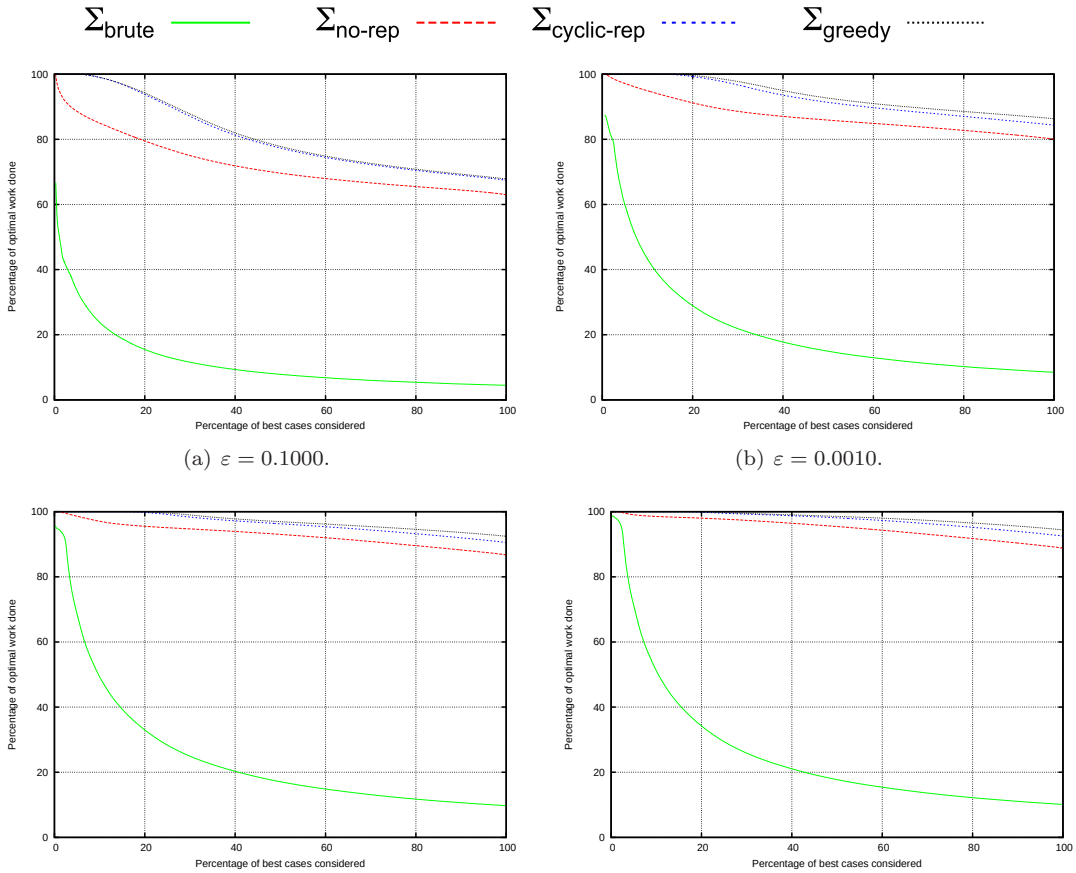
(b) 10 computers.

(c) 25 computers.

(d) 50 computers.

(e) 100 computers.

Figure 62: Experiment (E5) using different numbers of computers.



(a) $\varepsilon = 0.1000$.

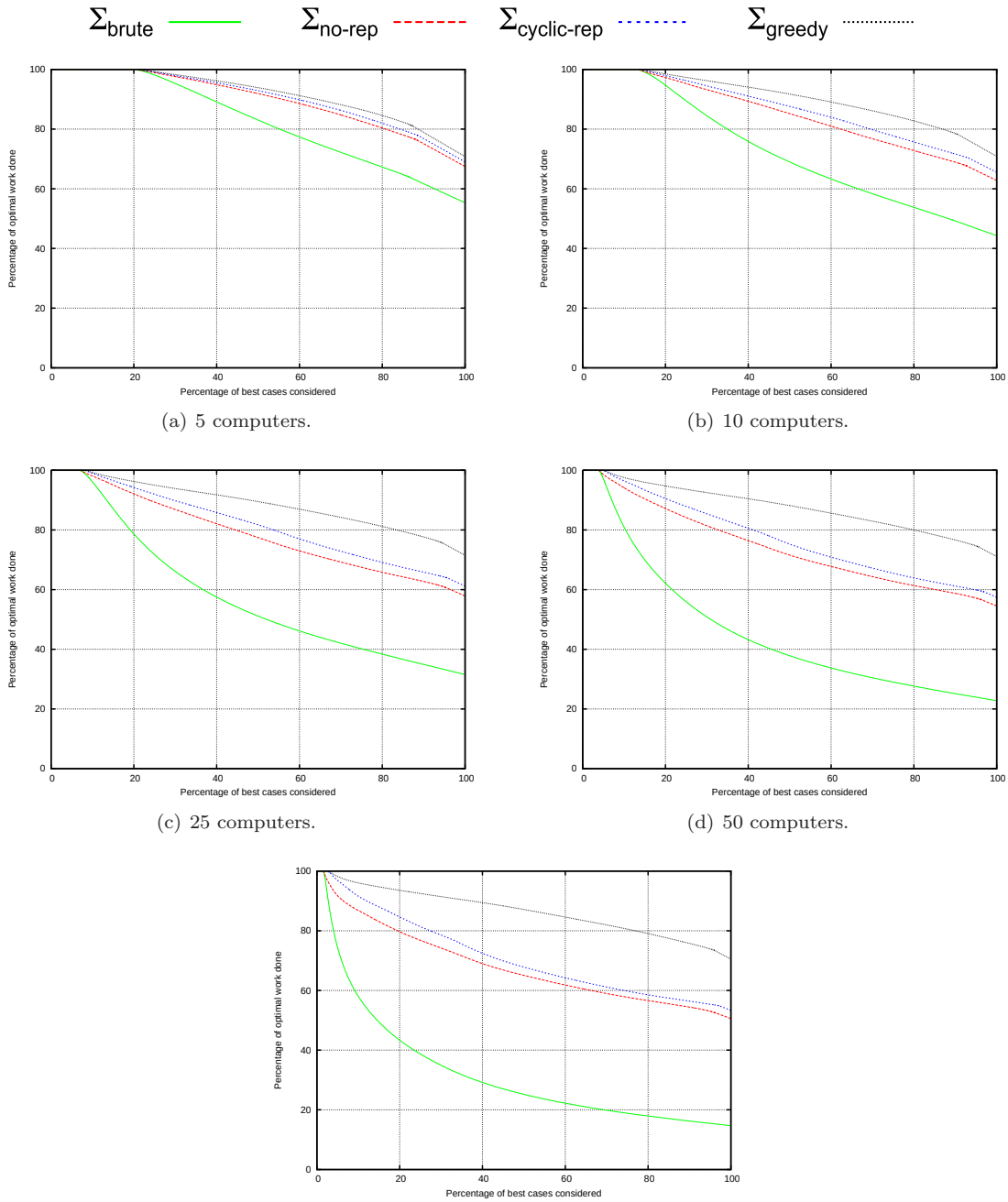
(b) $\varepsilon = 0.0010$.

(c) $\varepsilon = 0.0010$.

(d) $\varepsilon = 0.0001$.

Figure 63: Experiment (E5) using different values of start-up cost.

C Experiments with general risk functions



(a) 5 computers.

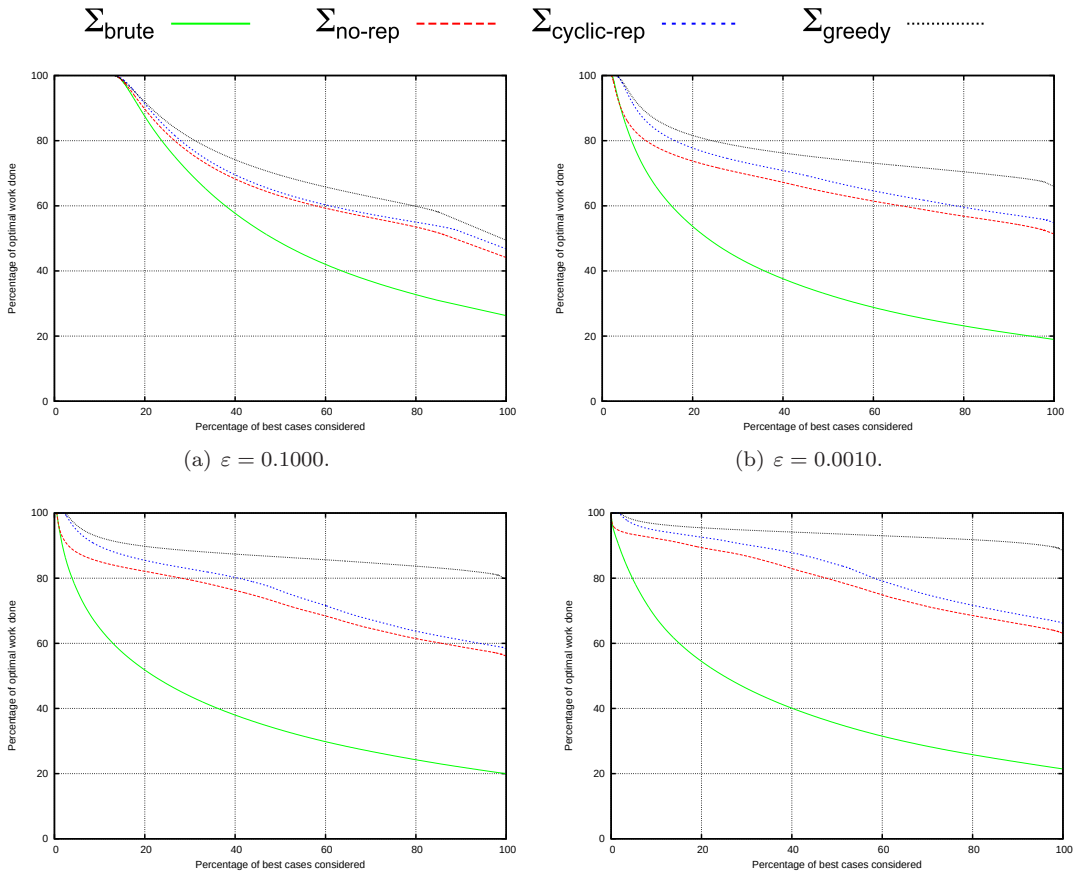
(b) 10 computers.

(c) 25 computers.

(d) 50 computers.

(e) 100 computers.

Figure 64: Experiments using different numbers of computers.



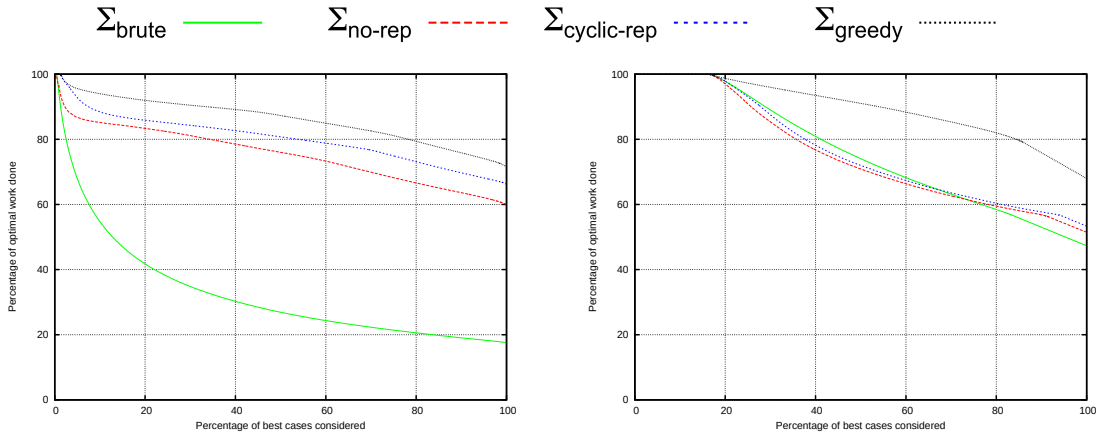
(a) $\varepsilon = 0.1000$.

(b) $\varepsilon = 0.0010$.

(c) $\varepsilon = 0.0010$.

(d) $\varepsilon = 0.0001$.

Figure 65: Experiments using different values of start-up cost.



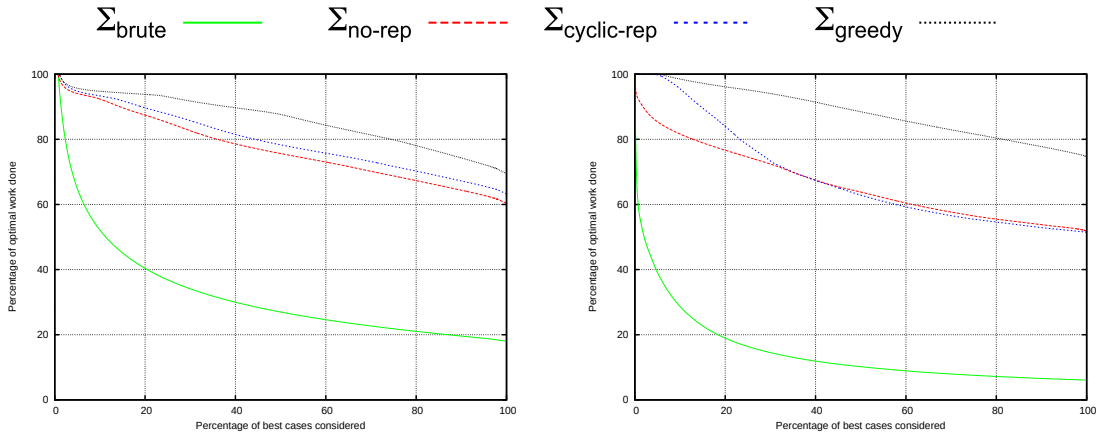
(a) Trace 0.

(b) Trace 1.

(c) Trace 2.

(d) Trace 3.

Figure 66: Experiments using different different traces (a).



(a) Trace 4.

(b) Trace 5.

(c) Trace 6.

(d) Trace 7.

Figure 67: Experiments using different different traces (b).